TOWARDS UNIFIYING MOBILITY DATASETS

A DISSERTATION SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

By
Fuat Basık
December 2019

TOWARDS UNIFIYING MOBILITY DATASETS

By Fuat Basık

December 2019

We certify that we have read this dissertation and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Özgür Ulusoy(Advisor)

Buğra Gedik(Co-Advisor)

Hakan Ferhatosmanoğlu(Co-Advisor)

İ. Sengör Altıngövde

A. Ercüment Çiçek

Engin Demir

Eray Tüzün

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

TOWARDS UNIFIYING MOBILITY DATASETS

Fuat Basık
Ph.D. in Computer Engineering
Advisor: Özgür Ulusoy
Co-Advisor: Buğra Gedik
Co-Advisor: Hakan Ferhatosmanoğlu
December 2019

With the proliferation of smart phones integrated with positioning systems and the increasing penetration of Internet-of-Things (IoT) in our daily lives, mobility data has become widely available. A vast variety of mobile services and applications either have a location-based context or produce spatio-temporal records as a byproduct. These records contain information about both the entities that produce them, as well as the environment they were produced in. Availability of such data supports smart services in areas including healthcare, computational social sciences and location-based marketing. We postulate that the spatio-temporal usage records belonging to the same real-world entity can be matched across records from different location-enhanced services. This is a fundamental problem in many applications such as linking user identities for security, understanding privacy limitations of location based services, or producing a unified dataset from multiple sources for urban planning and traffic management. Such integrated datasets are also essential for service providers to optimise their services and improve business intelligence. As such, in this work, we explore scalable solutions to link entities across two mobility datasets, using only their spatio-temporal information to pave to road towards unifying mobility datasets. The first approach is rule-based linkage, based on the concept of *k-l diversity* — that we developed to capture both spatial and temporal aspects of the linkage. This model is realized by developing a scalable linking algorithm called *ST-Link*, which makes use of effective spatial and temporal filtering mechanisms that significantly reduce the search space for matching users. Furthermore, *ST-Link* utilizes sequential scan procedures to avoid random disk access and thus scales to large datasets. The second approach is similarity based linkage that proposes a mobility based representation and similarity computation for entities. An efficient matching process is then developed to identify the final linked pairs, with an automated mechanism to decide when to stop the linkage. We scale the process with a locality-sensitive hashing (LSH)

based approach that significantly reduces candidate pairs for matching. To realize the effectiveness and efficiency of our techniques in practice, we introduce an algorithm called *SLIM*. We evaluated our work with respect to accuracy and performance using several datasets. Experiments show that both *ST-Link* and *SLIM* are effective in practice for performing spatio-temporal linkage and can scale to large datasets. Moreover, the LSH-based scalability brings two to four orders of magnitude speedup.

Keywords: Mobility Data, Data Integration, Spatio-Temporal Linkage, Scalability.

ÖZET

MOBİL VERİ KÜMELERİNİ BİRLEŞTİRMEYE DOĞRU

Fuat Basık
Bilgisayar Mühendisliği, Doktora
Tez Danışmanı: Özgür Ulusoy
İkinci Tez Danışmanı: Buğra Gedik
İkinci Tez Danışmanı: Hakan Ferhatosmanoğlu
Aralık 2019

Konumlandırma sistemleriyle entegre akıllı telefonların yaygınlaşması ve nesnelerin internetinin (Internet of Things - IoT) günlük hayatımızdaki etkisinin artmasıyla birlikte, mobil veri kümeleri yaygın bir şekilde erişilebilir oldu. Günümüzde birçok mobil servis ve uygulama, ya lokasyon bazlı bir içeriğe sahip ya da yan ürün olarak mekan-zaman bilgisi içeren kayıtlar üretmektedir. Bu kayıtlar, hem kendilerini üreten varlıklar veya kullanıcılar, hem de üretildikleri çevre hakkında bilgiler içerir. Bu kayıtların kullanılabilirliği sağlık hizmetleri, hesaplamalı sosyal bilimler ve konum tabanlı pazarlama gibi alanlarda akıllı hizmetleri destekler.

Bu çalışma, farklı servislerin kullanımı sonucu elde edilen, gerçek dünyada aynı varlık tarafından üretilen ve mekan-zaman bilgisi içeren kayıtların eşleştirilebileceğini öne sürmektedir. Bu eşleştirme, güvenlik için kullanıcı kimliklerini bağlama, konum tabanlı hizmetlerin gizlilik sınırlamalarını anlama ve kentsel planlama ve trafik yönetimi için birden fazla kaynaktan birleşik bir veri kümesi oluşturma gibi birçok uygulamada temel bir zorunluluktur. Bu tür birleştirilmiş mobil veri kümeleri, servis sağlayıcıların hizmetlerini optimize etmeleri ve iş zekasını geliştirmeleri için de önemlidir. Dolayısıyla, bu çalışma, iki mobil veri kümesindeki varlıkları birbirine bağlamak ve mobil veri kümelerini birleştirmeye giden yolda bir adım daha ilerleyebilmek amacıyla, yalnızca mekansal-zamansal bilgileri kullanarak ölçeklenebilir çözümler araştırmak için yapılmıştır ve sonuç olarak bu eşleştirmeye iki farklı yaklaşım önermektedir.

Önerilen ilk yaklaşım, kullanım kayıtları arasındaki yakınlığın hem mekansal hem de zamansal yönlerini kapsamak üzere geliştirilen, k - l *çeşitleme* kavramına dayanan *kurala dayalı eşlemedir*. Bu modelin etkinliği ve ölçeklenebilirliği, eşleşen

varlıklar için arama alanını önemli ölçüde azaltan etkili mekansal ve zamansal filtreleme mekanizmalarını kullanan *ST-Link* adlı ölçeklenebilir bir eşleme algoritması geliştirilerek ölçülmektedir. Bu algoritma, mekansal ve zamansal filtreleme adımlarına ek olarak rastgele disk erişimden kaçınan sıralı tarama prosedürlerini kullanarak büyük veri kümelerine ölçeklenmeyi arttırır.

İkinci yaklaşım, varlıkların mekan-zaman bilgisi içeren kullanım geçmişlerinin gösterimi ve bu gösterimler arasındaki benzerliğin tanımlanmasına bağlı, *benzerliğe dayalı eşleştirmedir*. Bu yaklaşım aynı zamanda eşleştirme işleminin ne zaman durduracağına otomatik olarak karar veren bir durma mekanizması ve eşleşen varlıkları tespit edebilmek için etkili bir eşleştirme sistemi geliştirmektedir. Büyük veri kümelerine ölçeklenebilirlik, eşleştirme sisteminin işleyeceği aday varlık çiftlerini önemli oranda azaltan yakınlığa-duyarlı-karım (Locality-Sensitive-Hashing LSH) sayesinde yapılmaktadır. Çalışma bu modelin ve yakınlığa-duyarlı-karım tabanlı ölçeklenebilirliğin etkinliğini ve verimliliğini ölçmek için *SLIM* adlı bir algoritma da içermektedir.

Çalışma son kısmında, hem kural tabanlı, hem de benzerlik tabanlı eşleme yaklaşımlarını çeşitli veri setleri kullanarak doğruluk ve performans açısından inceleyen deneysel değerlendirmeyi sunmaktadır. Bu deneyler, hem *ST-Link* hem de *SLIM* algoritmalarının, mekansal-zamansal eşleme için pratikte etkili olduğunu ve büyük veri kümelerine ölçeklenebileceğini göstermektedir. Dahası, yakınlığa-duyarlı-karım tabanlı ölçeklenebilirlik adımının eşleştirme işlemini 10^2 ila 10^4 kat hızlandırdığı gözlemlenmiştir.

Anahtar sözcükler: Mobil Veri Kümeleri, Data Kümelerinde Birleştirme, Mekan-Zaman Eşleştirmeleri, Ölçeklenebilirlik.

Acknowledgement

I would like to dedicate this thesis to two people. My mother and my wife. None of this would have been possible without their love.

First and foremost, I owe my deepest gratitude to my supervisors, Prof. Dr. Özgür Ulusoy, Prof. Dr. Hakan Ferhatosmanoğlu, and Assoc. Prof. Dr. Buğra Gedik for their encouragement, motivation, guidance and support throughout my studies.

I would like to thank my tracking commitee members, Assoc. Prof. Dr. İ. Sengör Altıngövde and Asst. Prof. Dr. A. Ercüment Çiçek. I would also like to thank Asst. Prof. Dr. Eray Tüzün and Asst. Prof. Dr. Engin Demir for kindly accepting to be in my committee. I owe them my appreciation for their support and helpful suggestions.

I would like to thank to my brother Uğur, his wife Seda, Hakan and Berrak for always being cheerful and supportive. I am grateful for all the selflessness and the sacrifices they have made on my behalf.

I consider myself to be very lucky to have the most valuable friends Caner, Anıl, Didem, Çağlar, Arif and Taylan, and thank them for sharing their knowledge and supporting me all the time. I would like to specifically thank Ebru and Damla for always being there for me.

Part of this work is reprinted, with permission, F. Basık and B. Gedik and C. Etemoğlu and H. Ferhatosmanoğlu, "Spatio-Temporal Linkage over Location-Enhanced Services," in IEEE Transactions on Mobile Computing, vol. 17, no. 2, pp. 447-460, 1 Feb. 2018.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the proliferation of smart phones integrated with positioning systems and the increasing penetration of Internet-of-Things (IoT) in our daily lives, mobility data has become widely available [1]. The size of the digital footprint left behind by entities interacting with these online services is increasing at a rapid pace, due to the popularity of location based services, social networks and related online services. An important portion of this footprint contains spatio-temporal references and is a fertile resource for applications for social good and business intelligence [2]. Availability of such data supports smart services in areas including healthcare [3], computational social sciences [4], and location-based marketing [5].

There are many examples of services that leave spatio-temporal footprint behind. Payments made by credit cards produce spatio-temporal records containing the time of the payment and the location of the store. Geotagging has become a common functionality in many content sharing and social network applications. Location sharing services such as Swarm[1] and social networks help people share their whereabouts with others. These records contain information about both the entities that produce them, as well as the environment they were produced in. We refer to the services that create spatio-temporal records of their usage as *Location Enhanced Services* (LES) and their data as *mobility data* or *location*

[1] `https://www.swarmapp.com`

based dataset.

We consider two varieties of location enhanced services based on an entity's level of involvement in the production of spatio-temporal usage records. Entities of *explicit location enhanced services* actively participate in sharing their spatio-temporal information. Location-based social network services, like Foursquare/Swarm, are well-known examples of such services, where the user explicitly checks-in to a particular point of interest at a particular time. On the other hand, *implicit location enhanced services* produce spatio- temporal records of usage as a byproduct of a different activity, whose focus is not sharing the location. For instance, when a user makes a payment with her credit card, a record is produced containing time of the payment and location of the store. Same applies for the cell phone calls, since originating cell tower location is known to the service provider.

There are several studies that analyze the dataset generated by a location enhanced service to model the mobility patterns and build applications with positive impact on the society, such as reducing traffic congestion, lowering noise/air pollution levels, and analyzing the spread of influenza using transportation networks [6, 7]. Most of the studies focus on a single dataset, which provides only a partial and biased state, failing to capture the complete patterns of mobility. To produce a comprehensive view of mobility, one needs to integrate multiple datasets, potentially from disparate sources. Such integration enables knowledge extraction that cannot be obtained from a single data source, and benefits a wide range of applications and machine learning tasks [8, 9]. A recent example is discovering regional poverty by jointly using mobile and satellite data in a developing country, where accurate demographic information is not available [10]. In another work, urban social diversity is measured by jointly modeling the attributes from Twitter and Foursquare [11].

Integration also helps to overcome with inaccuracy of information that is provided by a single dataset. Consider the simple query of counting the number of entities in a certain location at a given time. While each individual dataset provides a partial answer, a more complete and accurate answer can be reached by

integrating the results from multiple sources which are potentially overlapping. Therefore, spatio-temporal linkage is necessary to avoid over- or under-estimation of population densities using multiple sources of data, e.g., signals from wifi based positioning and mobile applications. This is essential to achieve the ambitious goal of producing a unified mobility dataset from multiple sources.

Spatio-temporal linkage solutions are useful in several other applications, such as user identification for security purposes, and understanding the privacy consequences of releasing mobility datasets [12]. An outcome of work such as ours is to help developing privacy advisor tools where location based activities are assessed in terms of their user identity linkage likelihood.

Identifying the matching entities across two mobility datasets is a non-trivial task, since some datasets are anonymized due to privacy or business concerns, and hence unique identifiers are often missing [13]. The linkage can be considered generic only if performed using only spatio-temporal attributes, as we target in this work. This helps to avoid the use of personally identifying information [14] and additional sensitive data, and simplifies the procedures to share data for social good and research purposes without having to expose personally identifying information. Consequently, anonymity assumption not only generalizes the linkage algorithms but also show that in addition to social good applications using integrated mobility data, they are also important in understanding privacy indications of anonymized data [12]. As such, in this work, we explore scalable solutions to link entities across two mobility datasets, using only their spatio-temporal information to pave the road towards unifying mobility datasets. There are several challenges associated with such linkage across mobility datasets.

First, unlike in traditional record linkage [15, 16, 17], where it is easier to formulate linkage based on a traditional similarity measure defined over records (such as Minkowski distance or Jaccard similarity), in spatio-temporal linkage similarity needs to be defined based on time, location, and the relationship between the two. For a pair of entities from two different datasets to be considered similar, their usage history must contain records that are close both in space and time. Equally importantly there must not be *negative matches*, such as records

that are close in time, but far in distance. We call such negative matches, *alibi*s.

Second, once similarity scores are assigned to entity pairs, an efficient matching process needs to identify the final linked pairs. A challenging problem is to automate the decision to stop the linkage to avoid false positive links. In a real-world setting, it is unlikely to have the entities from one dataset as a subset of the other, which is a commonly made assumption. Frequently, it is not even possible to know the intersection amount in advance. This is an important but so far overseen issue in the literature [18, 19, 20].

Third, performing the linkage in a reasonable amount of time is crucial, considering that the mobile services have millions of entities interacting with them everyday. Comparing each pair of entities would require quadratic number of similarity score computations. Avoiding this exhaustive search for matching pairs and focusing on those pairs that are likely to be matching can scale the linkage to a large number of entities.

In this dissertation, we present two novel scalable linkage approaches, called *ST-Link* and *SLIM* for finding the matching entities across two mobility datasets, relying on the spatio-temporal information. In both approaches, similarity of entity pairs is defined over aggregate similarities of records in their usage histories. They both do not penalize the score when one entity has activity in a particular time window but the other does not, but do penalize the existence of cross-dataset activities that are close in time but distant in space (aka *alibi*s [21, 13],). This is an essential property that supports mobility linkage. In *ST-Link*, the similarity of a pair of entities is defined over co-occurring records, both temporally and spatially. However, as we will detail later not all co-occurring records contribute fully and equally to the overall aggregation (Chapter 3). In fact, contribution of a co-occurring record pair to the overall aggregation depends on the uniqueness of this co-occurrence. In *SLIM* however, given an entity, we introduce a *mobility history* representation, by distributing the recorded locations over time-location bins (Section 4.1) and defining a novel similarity score for histories, based on a scaled aggregation of the proximities of their matching bins (Section 4.3.1). The proposed similarity definition provides several important properties. It awards

the matching of close time-location bins and incorporates the frequency of the bins in the award amount. It normalizes the similarity scores based on the size of the histories in terms of the number of time-location bins.

Once the similarity scores are computed, the second step is to define a *linkage model* or *matching process* to identify the matching entities. A particularly challenging step here is to decide on rule or a similarity score threshold to stop the linkage. To cope with this challenge *ST-Link* leverages a rule based approach by introducing a novel *linkage model* based on *k-l diversity* — a concept we developed to capture both spatial and temporal diversity aspects of the linkage. Informally, a pair of entities, one from each dataset, is called k-l diverse if they have at least k co-occurring records (both temporally and spatially) in at least l different locations. Furthermore, the number of alibi events of such pairs should not exceed a predefined threshold. Based on the distribution of the all k-l values of candidate pairs we develop an automatic detection of k and l values technique based on best trade-off point detection. Different than *ST-Link*, in the *SLIM* the matched entities are linked via an automated linkage thresholding (Section 4.3.3). In this linkage model, similarity scores are used as weights to construct a bipartite graph (Section 4.3.2) which is used for maximum sum matching. To compute the similarity score threshold to stop the linkage, we first fit a mixture model, e.g., Gaussian Mixture Model (GMM), with two components over the distribution of the edge weights selected by the maximum sum bipartite matching. One of these components aims to model the true positive links and the other one is for false positive links. We then formulate the expected precision, recall, and $F1$-score for a given threshold, based on the fitted model and select the threshold that provides the maximum $F1$-Score and use it to filter the results to produce the linkage.

Naïve record linkage algorithms that compare every pair of records take $\mathcal{O}(n^2)$ time [22], where n is the number of records. However, such a computation would not scale to large dataset sizes that are typically involved in LES. Considering that location-based social networks get millions of updates every day, processing of hundreds of days of data for the purpose of linkage would take impractically long amount of time.

In order to link entities in a reasonable time, the *ST-Link* algorithm uses two filtering steps before pairwise comparisons of candidate entities are performed to compute the final linkage. Taking advantage of the spatio-temporal structure of the data, *ST-Link* first distributes entities over coarse-grained geographical regions that we call *dominating grid cells*. Such grid cells contain most of the activities of their corresponding entities. For two entities to link, they must have a common dominating grid. Once this step is completed, the linkage is independently performed over each dominating grid cell. During the temporal filtering step, *ST-Link* uses a sliding window based scan to build candidate entity pairs, while also pruning this list as alibis are encountered for the current candidate pairs. It then performs a reverse scan to further prune the candidate pair set by finding and applying alibis that were not known during the forward scan. Finally, our complete linkage model is evaluated over candidate pairs of entities that remain following the spatial and temporal filtering steps. Pairs of entities that satisfy k-l diversity are linked to each other.

To address the same challenge in the similarity based approach, we employ *Locality Sensitive Hashing* (LSH) [23]. To apply LSH in our context, we make use of the *dominating grid cell* concept (Section 4.4) again. Given a mobility history, we construct a list of dominating grid cells to act as signatures. We next apply a banding technique, by dividing the signatures into b bands consisting of r rows, where each band is hashed to a large number of buckets. The goal is to come up with a setting such that signatures with similarity higher than a threshold t is hashed to the same bucket at least once. We only compute the similarity score for the mobility history pairs hashed to the same bucket. Our experimental evaluation (Section 6) shows that this technique brings two to four orders of magnitude speedup to linkage with a slight reduction in the recall.

The structure of the processing pipeline for both *linkage models* resembles two-step approach of entity resolution techniques where blocking/indexing applied first and then, the similar entities are compared in detail. However, the goal here is slightly different where instead of linking the records of a dataset to each other, we aim to link the owners of the records. Therefore, our similarity score computation and scalability techniques are not defined over records but their

owners.

In summary, this work makes the following contributions:

- *Model.* We introduce two novel spatio-temporal linkage models. The first one is based on the concept of k-l diversity for matching. In the second one we devise a summary representation for the mobility records of the entities and a method to compute similarity among these summaries. The similarity score we introduce captures the closeness in time and location, tolerates temporal asynchrony, and penalizes the *alibi* records.

- *Algorithm.* To realize the linkage models in practice we develop: i) *ST-Link* algorithm that applies spatial and temporal filtering techniques to prune the candidate entity pairs in order to scale to large datasets and also performs mostly sequential I/O to further improve performance, ii) *SLIM* algorithm for linking entities across two mobility datasets, which relies on maximum bipartite matching over a graph formed using the similarity scores. One of our contributions is to detect an appropriate score threshold to stop the linkage; a crucial step for avoiding false positives.

- *Scalability.* To scale linkage for large datasets, we define scalability techniques tailored to our linkage models. For the rule-based linkage we take advantage of the data structure and employ spatial and temporal filtering techniques. For the similarity based linkage we use *locality-sensitive hashing* and show that it brings a significant speedup. To the best of our knowledge, this is the first application of LSH in the context of mobility history linkage.

- *Evaluation.* We perform extensive experimental evaluation using four real-world datasets. We compare *ST-Link* with *SLIM* and *Swoosh* [24]. We also compare *SLIM* with two additional existing approaches, GM [19] and Pois [14], and show that it has superior performance in terms of accuracy and scalability.

The rest of this dissertation is organized as follows. Chapter 3 gives the details

of the rule based linkage, and Chapter4 presents the similarity based linkage. In Chapter 5, we extend the linkage to multiple datasets, discuss storage overhead in case of uncertain spatial information and also handling dynamic changes in data. In Chapter 6 we present our experimental evaluation and in Chapter 7 we make an extensive literature review. Lastly, Chapter 8 concludes this work.

Chapter 2

Preliminaries

In this section, we present preliminaries including the definition of the key concepts and the notation used throughout the dissertation.

2.1 Notation and Preliminaries

Location Datasets. We define a location dataset as a collection of usage records from a location-based service. We use $\mathcal{I}$ and $\mathcal{E}$ to denote the two location datasets from the two services, across which the linkage will be performed. While our focus in this work is on performing linkage across two datasets, extensions to multiple datasets is possible via pair-wise linkage.

Entities. Entities are real-world systems or users, whose usage records are stored in the location datasets. Throughout this work the terms user and entity will be used interchangeably. They are represented in the datasets with their ids. An id uniquely identifies an entity within a dataset. However, since ids are anonymized, they can be different across different datasets and cannot be used for linkage. The set of all entities of a location dataset is represented as $U_{\mathcal{E}}$, where the subscript represents the dataset. Throughout this work, we use $u \in U_{\mathcal{E}}$ and $v \in U_{\mathcal{I}}$ to represent two entities from the two datasets.

Records. Records, or events, are generated as a result of entities interacting with the location-based service. Each record is a triple $\{u, l, t\}$, where for a record $r \in \mathcal{E}$, $r.u$ represents the id of the entity associated with the record. Similarly, $r.l$ and $r.t$ represent the location and timestamp of the record, respectively. Record locations could be in the form of a region or a point. In the *ST-Link* the record location is in the form of a region. *SLIM* assumes that the record locations are given as points but it can be extended to datasets that contain record locations as regions, by copying a record into multiple cells within the mobility histories using weights. We explore this approach in our experimental evaluation.

2.2 Spatio-Temporal Linkage

With these definitions at hand, we can define the problem as follows. Given two location datasets, $\mathcal{I}$ and $\mathcal{E}$, the problem is to find a one-to-one mapping from a subset of the entities in the first set to a subset of the entities in the second set. This can be more symmetrically represented as a function that takes a pair of entities, one from the first dataset and a second from the other, and returns a Boolean result that either indicates a positive linkage (true) or no-linkage (false), with the additional constraint that an entity cannot be linked to more than one entity from the other dataset. A positive linkage indicates that the relevant entities from the two datasets refer to the same entity in real-life.

More formally, we are looking for a linkage function $\mathcal{M} : U_{\mathcal{E}} \times U_{\mathcal{I}} \rightarrow \{0, 1\}$, with the following constraint:

$$\begin{aligned} &\exists\, u, v \text{ s.t. } \mathcal{M}(u, v) = 1 \\ &\qquad \Rightarrow \forall u' \neq u, v' \neq v,\ \mathcal{M}(u', v) = \mathcal{M}(u, v') = 0 \end{aligned}$$

Since location datasets are collected from different services, the entities would only partially overlap. In a real world setting, the size of this overlap might not be known in advance. Even when it is known, finding all positive linkages is often

not possible, as some of the entities may not have enough records to establish linkage. This means that a good linkage function should provide high precision but the recall could be limited if the matching entities do not have enough records.

Assume there is an oracle $\mathcal{G} : U_{\mathcal{E}} \times U_{\mathcal{I}} \rightarrow \{0, 1\}$ that could be used as the ground truth. This function returns a Boolean result that either indicates that the two entities are the same (true) or they are different (false).

Precision is defined as:

$$P = \frac{|\{(u, v) \text{ s.t. } \mathcal{M}(u, v) = 1 \wedge \mathcal{G}(u, v) = 1\}|}{|\{(u, v) \text{ s.t. } \mathcal{M}(u, v) = 1\}|} \tag{2.1}$$

Recall is defined as:

$$R = \frac{|\{(u, v) \text{ s.t. } \mathcal{M}(u, v) = 1\}|}{|\{(u, v) \text{ s.t. } \mathcal{G}(u, v) = 1\}|} \tag{2.2}$$

Chapter 3

Rule Based Linkage

In this chapter we discuss the rule based linkage approach. We first give the details of the k-l diversity concept and define the matching process under this model. We then explain the ST-Link algorithm and describe how it implements the k-l diversity based spatio-temporal linkage in practice. We introduce spatial and temporal filtering steps, which help the ST-link to scale for large datasets. Lastly, we give the final linkage step and how to automatically define k and l values.

3.1 k-l Diversity

The core idea behind the rule based linkage model is to locate pairs of users whose events satisfy k-l diversity. Stated informally, a pair of users is called k-l diverse if they have at least k co-occurring events (both temporally and spatially) in at least l different locations. Furthermore, the number of alibi events of such pairs should not exceed a predefined threshold. In what follows we provide a number of definitions that help us formalize the proposed k-l diversity.

Co-occurrence. Two events from different datasets are called co-occurring if they are close in space and time. Eq. 3.1 defines the P relationship to capture

the closeness in space. For two records $i \in \mathcal{I}$ and $e \in \mathcal{E}$, P is defined as:

$$P(i, e) \equiv (i.r \cap e.r) \neq \emptyset, \tag{3.1}$$

where $i.r$ and $e.r$ are the regions of the two events. While we defined the closeness in terms of intersection of regions, other approaches are possible, such as the fraction of the intersection being above a threshold: $|i.r \cap e.r|/\min(|i.r|, |e.r|) \geq \epsilon$. Our methods are equally applicable to such measures.

Eq. 3.2 defines the T relationship to capture the closeness of events in time:

$$T(i, e) \equiv |i.t - e.t| \leq \alpha. \tag{3.2}$$

Here, we use the α parameter to restrict the matching events to be within a window of α time units of each other. Using Eq. 3.1 and Eq. 3.2, we define the *co-occurrence* function C as:

$$C(i, e) \equiv T(i, e) \wedge P(i, e) \tag{3.3}$$

Alibi. While a definition of similarity is necessary to link events from two different datasets, a definition of dissimilarity is also required to rule out pairs of users as potential matches in our linkage. Such *negative matches* enable us to rule out incorrect matches and also reduce the space of possible matches throughout the linkage process. We refer to these negative matches as *alibi*s.

By definition alibi means *"A claim or piece of evidence that one was elsewhere when an act is alleged to have taken place"*. In this work we use alibi to define events from two different datasets that happened around the same time but at different locations, such that it is not possible for a user to move from one of these locations to the other within the duration defined by the difference of the timestamps of the events. To formalize this, we define a *runaway* function R, which indicates whether locations of two events are close enough to be from the same user based on their timestamps. We define R as follows:

$$R(i, e) \equiv d(i.r, e.r) \leq \lambda \cdot |i.t - e.t| \tag{3.4}$$

Here, λ is the maximum speed constant and d is a function that gives the shortest distance between two regions. If the distance between the regions of two events is less than or equal to the distance one can travel at the maximum speed, then we cannot rule out linkage of users associated with these two events. Otherwise, and more importantly, these two events form an alibi, which proves that they cannot belong to the same user. Based on this, we define an alibi function, denoted by A, as follows:

$$A(i,e) \equiv T(i,e) \wedge \neg P(i,e) \wedge \neg R(i,e) \tag{3.5}$$

Entity linkage. The definitions we have outlined so far are on pairs of events, and with these definitions at hand, we can now move on to definitions on pairs of entities/users. Let $u \in U_{\mathcal{I}}$ and $v \in U_{\mathcal{E}}$ be two users. We use $\mathcal{I}_u$ to denote the events of user u and $\mathcal{E}_v$ to denote the events of user v. In order to be able to decide whether two users are the same entity or not, we need to define a matching between their events.

Initially, let us define the set of all co-occurring events of users u and v, represented by the function F. We have:

$$F(u,v) = \{(i,e) \in \mathcal{I}_u \times \mathcal{E}_v : C(i,e)\} \tag{3.6}$$

F is our *focus set* and contains pairs of co-occurring events of the two users. However, in this set, some of the events may be involved in more than one co-occurring pairs. We restrict the matching between the events of two users by disallowing multiple co-occurring event pairs containing the same events. Accordingly, we define $\mathcal{S}$ as the set containing all possible subsets of F satisfying this restriction. We call each such subset an *event linkage set*. Formally, we have:

$$\begin{aligned} \mathcal{S}(u,v) = \{S \subseteq F(u,v) : \\ \nexists\{(i_1,e_1),(i_2,e_2)\} \subseteq S \text{ s.t. } i_1 = i_2 \vee e_1 = e_2\} \end{aligned} \tag{3.7}$$

We say that the user pair (u,v) satisfy k-l diversity if there is at least one event linkage set $S \in \mathcal{S}(u,v)$ that contains k co-occurring event pairs and at least l of them are at different locations. However, each co-occurring event pair does not

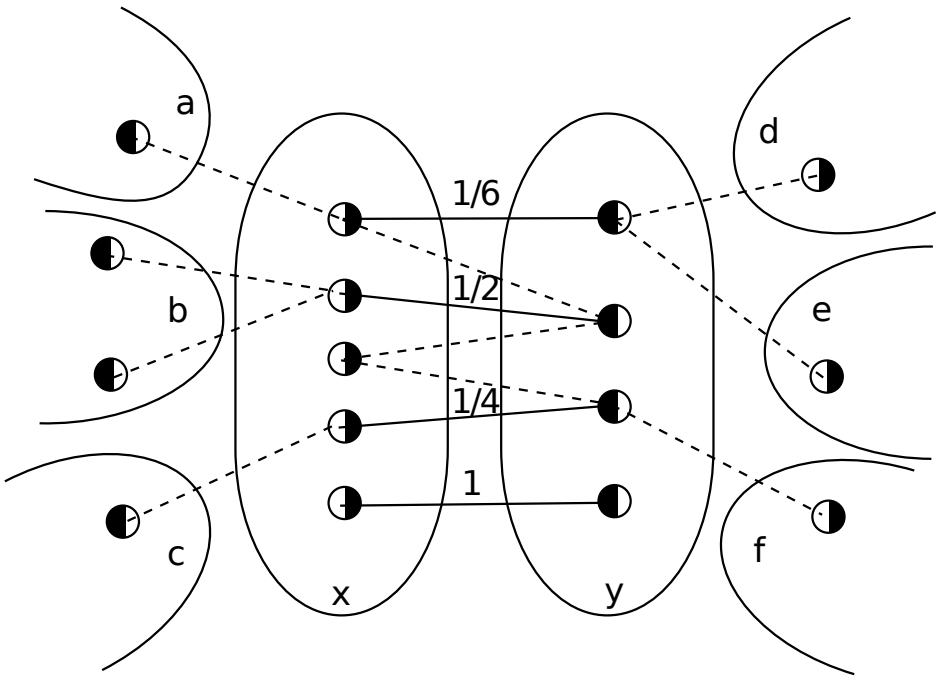


Figure 3.1: Sample event linkage set (solid lines) for users u and v. The co-occurring event pairs are shown using dashed lines. Events from a given user are shown within circles. Users a, b, c, and v are from one LES, and the users d, e, f, and u are from the other LES.

count as 1, since there could be many other co-occurring event pairs outside of S or even F that involve the same events. As such, we weight these co-occurring event pairs (detailed below). Figure 3.1 shows a sample event linkage set with weights for the co-occurring event pairs.

k co-occurring event pairs. Let S be an event linkage set in $\mathcal{S}(u, v)$ and let $\mathcal{C}$ be a function that determines whether the co-occurring event pairs in S satisfy the co-occurrence condition of k-l diversity. We have:

$$\mathcal{C}(S) \equiv \sum_{(i,e)\in S} w(i, e) \geq k \tag{3.8}$$

The weight of a co-occurring event pair is defined as:

$$\begin{aligned} w(i, e) =& |\{i_1.u : C(i_1, e) \wedge i_1 \in \mathcal{I}\}|^{-1} \cdot \\ & |\{e_1.u : C(i, e_1) \wedge e_1 \in \mathcal{E}\}|^{-1} \end{aligned} \tag{3.9}$$

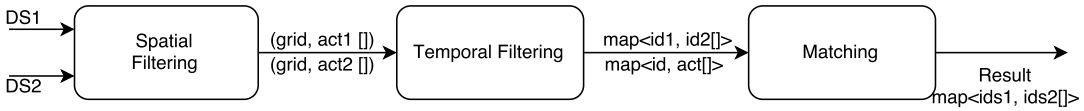


Figure 3.2: Data processing pipeline of ST-Link.

Here, given a co-occurring event pair between two users, we check how many possible users' events could be matched to the these events. For instance, in the figure, consider the solid line at the top with the weight 1/6. The event on its left could be matched to events of 2 different users, and the event on its right could be matched to events of 3 different users. To compute the weight of a co-occurring pair, we multiply the inverse of these user counts, assuming the possibility of matching from both sides are independent. As such, in the figure, we get $1/2 \cdot 1/3 = 1/6$.

l diverse event pairs. For $S \in \mathcal{S}$ (u, v) to be l-diverse, there needs to be at least l unique locations for the co-occurring event pairs in it. However, for a location to be counted towards these l locations, the weights of the co-occurring event pairs for that location must be at least 1. Let $\mathcal{D}$ denote the function that determines whether the co-occurring event pairs in S satisfy the diversity condition of k-l diversity. We have:

$$\mathcal{D}(S) \equiv |\{p \in \mathcal{P} : \sum_{\substack{(i,e)\in S \text{ s.t.} \\ p \cap i.r \cap e.r \neq \emptyset}} w(i, e) \geq 1\}| \geq l \tag{3.10}$$

Here, one subtle issue is defining a unique location. In Eq. 3.10 we use $\mathcal{P}$ as the set of all unique locations. This could simply be a grid-based division of the space. In our experiments, we use the regions of the Voronoi diagram formed by cell towers as our set of unique locations.

Before we can formally state the k-l diversity based linkage, we have to define the alibi relation for user pairs. Let $\mathcal{A}$ denote a function that determines whether there are more than a alibi events for a given pair of users. Intuitively, having a single alibi is enough to decide that user u and v are not the same entity, but when there is inaccurate information, disregarding candidate pairs with a single

alibi event might lead to false negatives. We have:

$$\mathcal{A}(u,v) \equiv |i, e \in \mathcal{I}_u \times \mathcal{E}_v, s.t. A(i,e)| \leq a \tag{3.11}$$

With these definitions at hand, finally we can define the spatio-temporal linkage function L from the original problem formulation from Chapter 2 that determines whether users u and v satisfy k-l diversity as follows:

$$L(u,v) \equiv \neg\mathcal{A}(u,v) \ \wedge \ S \in \mathcal{S}(u,v) \ s.t. \ (\mathcal{C}(S) \wedge \mathcal{D}(S)) \tag{3.12}$$

Finally, the linkage function $\mathcal{M} : U_{\mathcal{E}} \times U_{\mathcal{I}} \rightarrow \{0,1\}$ from the original problem formulation from Chapter 2 can be defined to contain only matching pairs of users based on L, such that there is no ambiguity. Formally:

$$\mathcal{M}(u,v) = \begin{cases} 1 & L(u,v) \ \wedge \ \forall u' \neq u, v' \neq v, \ L(u',v) = L(u,v') = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

3.2 Example Scenario

Consider three colleagues Alice, Bob, and Carl who are working in the same office. Assume that they all use two *LES*s: *les1* and *les2*. Both services generate spatio-temporal records only when they are used. The service provider would like to link the profiles of users common in both services. However, Bob uses the services only when he is at the office. On the other hand, Alice and Carl use the services frequently while at work, at home, and during vacations. Let us also assume that Alice and Carl live on the same block, but they take vacations at different locations.

When records of Alice from *les1* are processed against records of Carl from *les2*, we will encounter *co-occurrences* with some amount of diversity, as they will have matching events from work and home locations. However, we will encounter alibi events during vacation time. In this case, *alibi* checks will help us rule out the match.

When records of Alice from *les1* are processed against records of Bob from *les2*, the number of *co-occurrence*s will be high, as they are working in the same office. Yet, *diversity* will be low, as Bob does not use the services outside of the office. This also means we will not encounter any *alibi* events with Alice. In this case, diversity will help us rule out the match.

In contrast to these cases, when Alice's own usage records from *les1* and *les2* are processed, the resulting *co-occurrences* will contain high diversity since Alice uses the services at work, home, and during vacations, and will contain no *alibi*s.

In this example scenario, high number of *co-occurrences* helped us distinguish between mere coincidences and potential candidate pairs. The *alibi* definition helped us to eliminate a false link between Alice and Carl. Finally, *diversity* helped us to eliminate a false link between Alice and Bob, even in the absence of alibi events.

3.3 ST-Link

In this section, we describe how the *ST-Link* algorithm implements k-l diversity based spatio-temporal linkage in practice. At a high-level, *ST-Link* algorithm performs filtering to reduce the space of possible entity matches, before it performs a more costly pairwise comparison of entities according to the formalization given in Section 3.1. The filtering phase is divided into two steps: *temporal filtering* and *spatial filtering*. The final phase of pairwise comparisons is called *linkage*.

3.3.1 Overview

Naïve algorithms for linkage repeatedly compare pairwise records, and thus take $\mathcal{O}(n^2)$ [22] time, where n is the number of records. Such algorithms do not scale to large datasets. To address this issue, many linkage algorithms introduce some form of pruning, typically based on blocking [25, 26, 27] or indexing [28, 29].

Identifying the candidate user pairs on which the full linkage algorithm is to be run can significantly reduce the complexity of the end-to-end algorithm. Accordingly, *ST-Link* algorithm incorporates pruning strategies, which are integrated into the spatial filtering and temporal filtering steps.

Figure 3.2 shows the pipelined processing of the *ST-Link* algorithm. Given two sources of data for location-enhanced services (DS1 and DS2 in the figure), the spatial filtering step maps users to coarse-grained geographical grid cells that we call *dominating grid cells*. Such cells contain most activities of the corresponding entities. Once this step is over, the remaining steps are independently performed for each grid.

The temporal filtering step slides a window over the time ordered events to build a set of candidate entity pairs. During this processing, it also prunes as many entity pairs as possible based on alibi events. As we will detail later in this section, a reverse window based scan is also performed to make sure that all relevant alibis are taken into account.

Following the spatial and temporal filtering steps, the complete linkage is performed over the set of candidate entity pairs. With a significantly reduced entity pair set, the number of compared events decreases significantly as well. Given two datasets $\mathcal{I}$ and $\mathcal{E}$, the linkage step calculates list of all matching pairs of entities as given in Eq. 3.13 without considering all possible entity pairs.

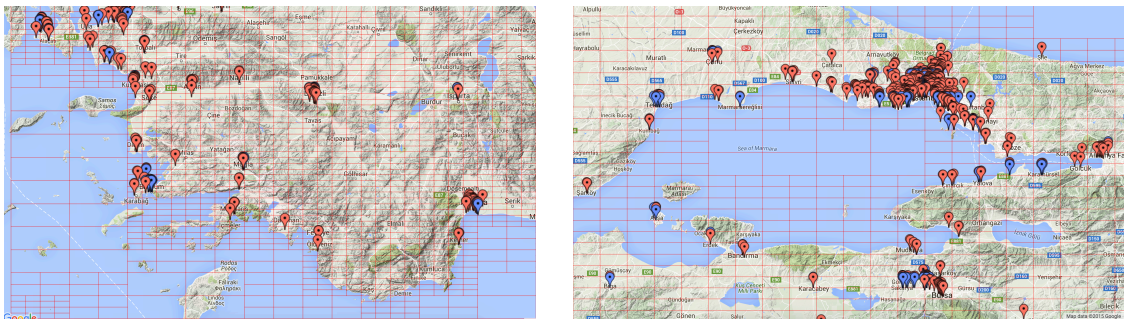
3.3.2 Spatial Filtering

By their nature, spatio-temporal data are distributed geographically. Spatial filtering step takes advantage of this, by partitioning the geographical region of the datasets into coarse-grained grid cells using quad trees [30]. Each entity is assigned to one (an in rare cases to a few) of the grid cells, which becomes that entity's dominating grid. The dominating grid of an entity is the cell that contains the most events from the entity. Entities that do not share their dominating grid cells are not considered for linkage. The intuition behind this filtering step is

that, if entity u from dataset $\mathcal{E}$ and entity v from dataset $\mathcal{I}$ are the same real world entity, then they are expected to have the same dominating grid cells.

3.3.2.1 Coarse Partitioning

For quad-tree generation in the *ST-Link* algorithm, we continue splitting the space until the grid cells size hits a given minimum. For our experiments, we make sure that the area of the grid cells is at least 100 km squares. For users, the grid cells should be big enough to cover a typical user's mobility range around his home and work location. If the minimum grid cell size is too small, then the spatial filtering can incorrectly eliminate potential matches, as the dominating grids from different datasets may end up being different. A concrete example is a user that checks in to coffee shops and restaurants around his work location, but uses a location-based match-making application only when he is at home.



(a) Grid cells. (b) Most popular venues.

Figure 3.3: Grids cells and top 1K venues

We also do not split grid cells that do not contain any events. As a result, not all grid cells are the same size. Figure 3.3 shows the grid cells for two selected areas in Turkey and the top 1K venues in those areas in terms of check-in counts, based on Foursquare check-in data.

3.3.2.2 Determining Dominating Grids

The determination of the dominating grid for an entity is simply done by counting the entity's events for different cells and picking the cell with the highest count. A subtle issue here is about entities whose events end up being close to the border areas of the grid cells. As a specific example, consider a user who lives in one cell and works in another. In this case, it is quite possible that a majority of the user's check-ins happen in one cell and the majority of the calls in another cell. This will result in missing some of the potential matches. To avoid this situation, we make two adjustments:

• If an event is close to the border, then it is counted towards the sums for the neighboring cell(s)[1] as well. We use a strip around the border of the cell to determine the notion of 'close to the border'. The width of the strip is taken as the 1/8th of the minimum cell's edge width. This means that around 43% of a grid overlaps with one or more neighboring grids. This adjustment resembles the loose quad trees [31].

• An entity can potentially have multiple dominating grid cells. We have found this to be rare for users in practice.

Figure 3.3b shows the resulting grids over selected areas in Turkey, and the most popular venues from our dataset. Red pins are showing the venues and the blue ones are showing the ones that count towards neighboring grids.

3.3.2.3 Forming Partitioned Datasets

Once the dominating grid cells of users are determined, we create grid cell specific datasets. For a given grid cell c, we take only the events of the entities who has c as a dominating grid cell. These events may or may not be in the grid cell c. Determination of the dominating grid cells of entities requires a single scan over the time sorted events from entities. The forming of the partitioned datasets requires a second scan.

[1] An event can count towards at most 3 neighbors, in case it is at the corner of the grid.

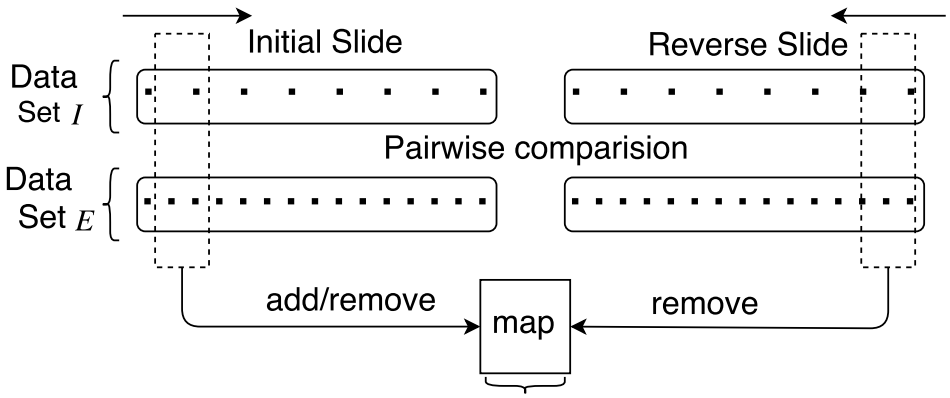


Figure 3.4: Temporal Filtering

3.3.3 Temporal Filtering

Temporal filtering aims at creating a small set of candidate user pairs on which the full linkage algorithm can be executed. To create this set, temporal filtering looks for user pairs that have co-occurring events, as expressed by Eq. 3.3. Importantly, temporal filtering also detects alibi events, based on Eq. 3.5, and prevents user pairs that have such alibi events from taking part in the candidate pair set.

Temporal filtering is based on two main ideas. First, a temporal window is slided over the events from two different datasets to detect user pairs with co-occurring events. Since co-occurring events must appear within a given time duration, the window approach captures all co-occurring events. Second, as the window slides, alibi events are tracked to prune the candidate user pair set. However, since the number of alibis is potentially very large, alibis are only tracked for the user pairs that are currently in the candidate set. This means that some relevant alibis can be missed if the user pair was added into the candidate set after an alibi event occurred. To process such alibis properly, a reverse window scan is performed, during which no new candidate pairs are added, but only alibis are processed. Algorithm 1 gives the pseudo-code of temporal filtering.

3.3.3.1 Data Structures

A window of size α (see Eq. 3.2) is slided jointly over both time sorted datasets. Figure 3.4 depicts this visually. Each time the window slides, some events from both datasets may enter and exit the window. We utilize two types of data structures to index the events that are currently in the window. The first type of index we keep is called the *user index*, denoted by UI_x, where $x \in \{\mathcal{I}, \mathcal{E}\}$. In other words, we keep separate user indexes for the two datasets. UI_x is a hash map indexed by the user. $UI_x[u]$ keeps all the events (from dataset x) of user u in the window. As we will see, this index is useful for quickly checking alibis.

The second type of index we keep is called the *spatial index*, denoted by LS_x, where $x \in \{\mathcal{I}, \mathcal{E}\}$. Again, we keep separate indexes for the two datasets. LS_x could be any spatial data structure like R-trees. $LS_x.query(r)$ gives all events whose region intersect with region r. As we will see, this index is useful for quickly locating co-occurring events.

In addition to these indexes, we maintain a global candidate set CS and a global alibi set AS. For a user u (from either dataset, assuming user ids are unique), $CS[u]$ keeps the current set of candidate pair users for u; and $AS[u]$ keeps the current known alibis users for u. It is important to note that AS is not designed to be exhaustive. For a user u, $AS[u]$ only keeps alibi users that have co-occurring events with u in the dataset.

3.3.3.2 Processing Window Events

The algorithm operates by reacting to events being inserted and removed from the window as the window slides over the dataset. As a result, an outermost while loop that advances the window until the entire dataset is processed. At each iteration, we get a list of events inserted ($N_{\mathcal{I}}^+$ and $N_{\mathcal{E}}^+$) and removed ($N_{\mathcal{I}}^-$ and $N_{\mathcal{E}}^-$) from the window. We first process the removed events, which consists of removing them from the spatial and user indexes. We then process the inserted events. We first process $N_{\mathcal{I}}^+$ against $UI_{\mathcal{E}}$ and $LS_{\mathcal{E}}$, then insert the events in $N_{\mathcal{I}}^+$

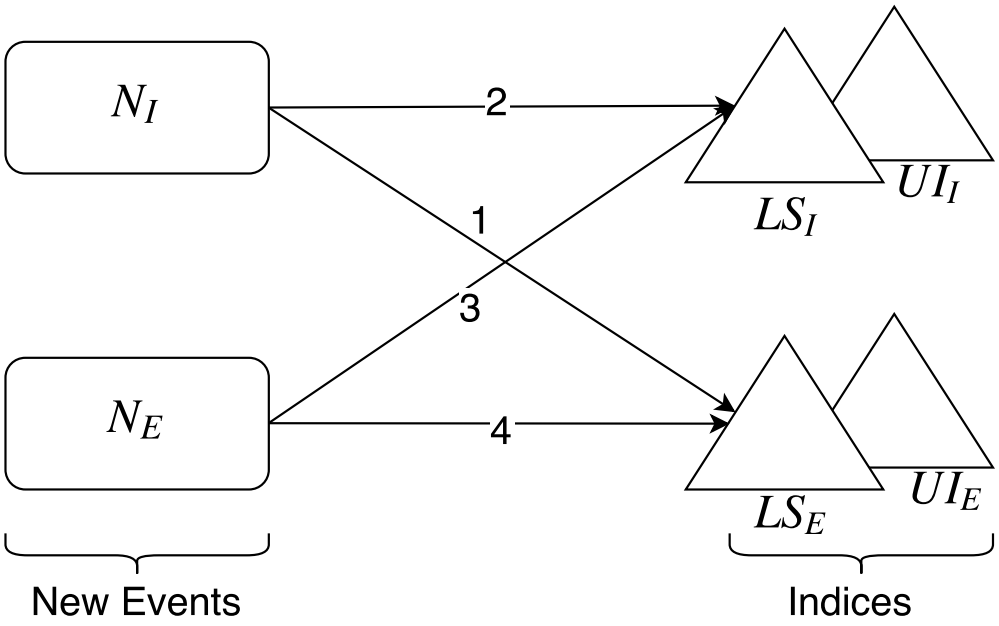


Figure 3.5: Calculation of the candidate set.

into $UI_{\mathcal{I}}$ and $LS_{\mathcal{I}}$, then process $N_{\mathcal{E}}^{+}$ against $UI_{\mathcal{I}}$ and $LS_{\mathcal{I}}$, and finally insert the events in $N_{\mathcal{E}}^{+}$ into $UI_{\mathcal{E}}$ and $LS_{\mathcal{E}}$. This ensures that all the events are compared, and no repeated comparisons are made. Figure 3.5 depicts the order of events visually.

To compare a new event i from dataset x against the events from dataset $\bar{x}$ that are already indexed in the window (where $\{x, \bar{x}\} = \{\mathcal{E}, \mathcal{I}\}$), we use the indexes $UI_{\bar{x}}$ and $LS_{\bar{x}}$. First, we find events that co-occur with i by considering events e in $LS_{\bar{x}}.query(i.r)$. These are events whose regions intersect with that of i. If the user of such an event e is not already a known alibi of the user of i (not in $AS[i.u]$) and if the co-occurrence condition $C(i, e)$ is satisfied, then the user $e.u$ and user $i.u$ are added as candidate pairs of each other. Second, and after all the co-occurrences are processed, we consider all candidate users of the event i's user, that is $CS[i.u]$, for alibi processing. For each user u in this set, we check if any of its events result in an alibi. To do this, we iterate over user u's events with the help of the index $UI_{\bar{x}}$. In particular, for each event e in $UI_{\bar{x}}[u]$, we check if i and e are alibis, using the condition $A(i, e)$. If they are alibis, then we remove u and i's user ($i.u$) from each other's candidate sets, and add them to their alibi

sets.

This completes the description of the forward scan of the window. An important point to note is that, during the forward window scan, we only check alibis for user pairs that are in the set of candidate pairs. It is possible that there exists an alibi event pair for users x and y, that appears before the first co-occurring event pair for these users. In such a case, during the processing of the alibi events we won't have this pair of users in our candidate set and thus their alibi will be missed. To fix this problem, we perform a reverse scan. During the reverse scan, we only process alibis, as no new candidate pairs can appear. Furthermore, we need to process alibi events for a user pair only if the events happened before the time this pair was added into the candidate set. For brevity, we do not show this detail in Algorithm 1. At the end of the reverse scan, the set CS contains our final candidate user pairs, which are sent to the linkage step. Temporal filtering is highly effective in reducing the number of pairs for which complete linkage procedure is executed. The experimental results show the effectiveness of this filtering.

When there is inaccurate information in the datasets, disregarding candidate pairs due to only a single alibi event might lead to false negatives. However, the algorithm is easily modifiable to use a threshold for alibi values. In this modified version, we update the structure of the alibi set AS to keep the number of alibi events of a pair as well. Now $AS[u]$ keeps the current known alibi users of user u with alibi event counts for each. Just like in the original algorithm, when two events i and e are compared we first check if the number of alibi events of users $i.u$ and $e.u$ exceeds the threshold. To avoid double counting, we reset the counters before the reverse scan. Since all alibi events of current candidate pairs will be counted in reverse scan, candidates whose count of alibi events exceed threshold will not be included in the resulting candidate set CS.

So far we have operated on time sorted event data and our algorithms used only sequential I/O. However, during the linkage step, when we finally decide whether a candidate user pair can be linked, we will need the time sorted events of the users at hand. For that purpose, during the forward scan, we also create

a disk-based index sorted by the user id and event time. This index enables us to quickly iterate over the events of a given user in timestamp order, which is an operation used by the linkage step. For this purpose, we use LevelDB [32] as an index, which is a log-structured merge-tree supporting fast insertions.

While writing the event to the disk-based index, we also include information about the number of unique users the event has matched throughout its stay in the forward scan window. This information is used as part of the weight calculation (recall Eq. 3.9) in the linkage step.

3.3.3.3 Handling Time Period in Events

The temporal filtering step scans time-ordered events by sliding a window of size α over them. This operation assumes that the time information is a point in time. Yet, there could be scenarios where the time information is a period (e.g., a start time and a duration). However, frequently, these records contain only start location of the event. For example, although Call Detail Records (CDR) have the start time of the call and the duration, they usually contain only the originating cell tower information. Considering mobility of the users, assuming a fixed location during this period would lead to location ambiguity.

If we have events with time periods and accurate location information is present during this period, we can adapt our approach to handle this. In particular, we need to avoid false negative candidate pairs when the event contains a time period. Since events are processed via windowing, making sure that the event with the time period information stays in the window as long as its time period is valid would guarantee that all co-occurrences will be processed. This requires creating multiple events out of the original event, with time information converted into a point in time and the correct location information attached to it. The number of such events is bounded by the time duration divided by the window size, α.

3.3.4 Linkage

The last step of the *ST-Link* algorithm is the linkage of the entities that are determined as possible pairs as a result of spatial and temporal filtering. This linkage is a realization of the k-l diversity based linkage model introduced in Section 3.1. Given two entities from two datasets, the linkage step uses the events of them to determine whether they can be linked according to Eq. 3.12. Thanks to efficient filtering steps applied on the data beforehand, the number of entity pairs for which this linkage computation is to be performed is significantly reduced.

For each entity pair, their events are retrieved from the disk-based index created as part of the forward scan during the temporal filtering. These events are compared for detecting co-occurring events. Co-occurring events are used to compute the k value, via simple accumulation of the co-occurrence weights. They are also used to accumulate weights for the places where co-occurring events occur. This helps us compute the l value, that is diversity. After all events of a pair of entities are compared, we check if they satisfy the k-l diversity requirement. Note that, it is not possible to see an alibi pair event at this step, as they are eliminated by the temporal filtering step.

There are a number of challenges in applying the k-l-diversity based linkage. The first is to minimize the number of queries made to the disk-based index to decrease the I/O cost. Events from the same entity are stored in a timestamped order within the index, which makes this access more efficient. Also, if one of the datasets is more sparse than the other, then the linkage can be performed by iterating over the entities of the dense datasets first, making sure their events are loaded only once. This is akin to the classical join ordering heuristic in databases.

Another challenge is the definition of the place ids to keep track of diversity. A place id might be a venue id for a Foursquare dataset, store id for credit card payment records, cell tower id for Call Detail Records, or a geographic location represented as latitude and longitude. An important difference is the area of coverage for these places. Consider two datasets of Foursquare check-ins and

Call Detail Records, and places based on venues. If a user visits several nearby coffee shops and makes check-ins and calls, these will be considered as diverse even though they are not geographically diverse. The use of cell tower coverage areas is a more practical choice for determining places.

The last challenge is about matching events. Recall from Figure 3.1 that events of two entities can be matched in multiple different ways, resulting in different weights for the co-occurrences. Ideally, we want to maximize the overall total weight of the matching, however this would be quite costly to compute, as the problem is a variation of the *bipartite graph assignment problem*. As a result, we use a greedy heuristic. We process events in a timestamped order and match them to the co-occurring event from the other entity that provides the highest weight. Once a match is made, event pairs are removed from the dataset so that they are not re-used.

Different k-l value pairs may perform significantly different in terms of precision and recall, depending on the frequencies of the events in the datasets. An ad-hoc approach is to decide the k and l values based on observation of results from multiple experimental runs. A more robust technique we used is to detect the best *trade-off* point (a.k.a *elbow point*) on a curve. Given the *co-occurrence* and *diversity* distributions, we independently detect the elbow point of each, and set the k and l values accordingly. Although there is no unambiguous solution for detecting an elbow point, the maximum absolute second derivative is an approximation. Let A be an array of co-occurrence (or diversity) values with size n. Second derivative, SD, of point at index i can be approximated with a central difference as follows:

$$SD[i] = A[i+1] + A[i-1] - 2 * A[i] \tag{3.14}$$

The value at index $A[i]$, such that i has the maximum absolute $SD[i]$ value, is selected as the *elbow point* and k (or l) value is set accordingly.

Elbow detection technique is a simple yet effective technique to automate decision of k and l values. In our experiments, we were able to detect values that

give the best precision and recall, in multiple datasets, even when the frequencies of datasets differ dramatically.

Algorithm 1: Candidate Set Calculation

Data: $SR_{\mathcal{I}}$, $SR_{\mathcal{E}}$: Time sorted datasets of events
Result: CS: A set of candidate user pairs

```
CS ← ∅ ;     // Candidates, CS[u] is the list of pair users of u
AS ← ∅ ;     // Alibis, AS[u] is the list of alibi users of u
UI_x ← ∅, x ∈ {I, E} ;                    // User index over window
// UI_x[u]: events from x in window belonging to user u
LS_x ← ∅, x ∈ {I, E} ;                   // Spatial index over window
// LS_x.query(e.r): events from x in window intersecting event e
W ← window(SR_I, SR_E, α) ;               // Window over the datasets
// Forward scan phase
while W.hasNext() do                  // While more events after window
    // Get events inserted into and removed from the window
    (N_I^+, N_E^+, N_I^-, N_E^-) ← W.next()
    for x ∈ {I, E} do                                  // In both directions
        for i ∈ N_x^- do                             // For each removed event
            LS_x.remove(i.r, i) ;                  // Remove from spatial index
            UI_x[i.u] ← UI_x[i.u] \ i ;             // Remove from user index
    for (x, x̄) ∈ {(I, E), (E, I)} do                   // In both directions
        for i ∈ N_x^+ do                            // For each inserted event
            // Query spatially close elements
            for e ∈ LS_x̄.query(i.r) do
                if e.u ∉ AS[i.u] then                // If users are not alibi
                    if C(i, e) then                        // If events co-occur
                        // Add to the candidate set
                        CS[i.u] ← CS[i.u] ∪ {e.u}
                        CS[e.u] ← CS[e.u] ∪ {i.u}
            for u ∈ CS[i.u] do                      // For each candidate user
                // For each event of the user in the window
                for e ∈ UI_x̄[u] do
                    if A(i, e) then                   // If i and e is an alibi
                        // Add to the alibi set
                        AS[i.u] ← AS[i.u] ∪ {u}
                        AS[u] ← AS[u] ∪ {i.u}
                        // Remove from the candidate set
                        CS[i.u] ← CS[i.u] \ {u}
                        CS[u] ← CS[u] \ {i.u}
            LS_x.insert(i.r, i) ;                    // Add to spatial index
            UI_x[i.u] ← UI_x[i.u] ∪ {i} ;              // Add to user index
```

```
    // Reverse scan phase
    W ← reverse_window(SR_I, SR_E, α) ;     // Reverse sliding window
    while W.hasNext() do          // While more events after window
        (N_I^+, N_E^+, N_I^-, N_E^-) ← W.next()
        for x ∈ {I, E} do                          // In both directions
            for i ∈ N_x^- do                    // For each removed event
                UI_x[i.u] ← UI_x[i.u] \ i ;      // Remove from user index
        for (x, x̄) ∈ {(I, E), (E, I)} do           // In both directions
            for i ∈ N_x^+ do                   // For each inserted event
                for u ∈ CS[i.u] do             // For each candidate user
                    if A(i, e) then             // If i and e is an alibi
                        // Add to the alibi set
                        AS[i.u] ← AS[i.u] ∪ {u}
                        AS[u] ← AS[u] ∪ {i.u}
                        // Remove from the candidate set
                        CS[i.u] ← CS[i.u] \ {u}
                        CS[u] ← CS[u] \ {i.u}
            UI_x[i.u] ← UI_x[i.u] ∪ {i} ;          // Add to user index
    return CS ;                          // Return the candidate set
```

Chapter 4

Similarity Based Linkage

In this chapter, we present our similarity based linkage approach for mobility datasets. We first give the definition of mobility histories followed by a high level solution overview. We then define the similarity score computation of mobility histories and how to detect similarity score threshold to stop the linkage automatically. Based on the definition of similarity of mobility histories, we introduce how to employ locality sensitive hashing to reduce the number of pairs of mobility histories to compute. We conclude this chapter with an illustrative example and running time analysis of the linkage process.

4.1 Mobility Histories

We organize the records in a location dataset into *mobility histories*. Given an entity in a location dataset, its mobility history consists of an aggregated collection of its records from the dataset. Due to the aggregation, the mobility history of an entity is not as low-level as a trajectory, but instead more sparse in both the temporal and the spatial domains.

The location and time information contained in the records from the two location datasets can naturally be at different levels of granularity. Depending on the

use case, different applications might require different levels of granularity as well. Consider a navigation service requiring much higher location granularity than a facility recommendation application. Therefore, representation of the mobility history should be generic enough to capture the differences in spatio-temporal heterogeneity of the datasets to be linked. To address this need, we propose a hierarchical representation for mobility histories, as illustrated in Fig. 4.1. This representation distributes the records over time-location bins.

In the temporal domain, the data is split into time windows which are hierarchically organized as a tree to enable efficient computation of aggregate statistics. The leaves of this tree hold a set of spatial cell ids. A cell id is present at a leaf node if the entity has at least one record whose spatial location is in that cell and the record timestamp is in the temporal range of the window. Each non-leaf node keeps the occurrence counts of the cell ids in its sub-tree. The space complexity of this tree is similar to a segment tree, $\mathcal{O}(|\mathcal{E}| + |\mathcal{I}|)$. As we will detail in Sec. 4.4, the information kept in the non-leaf nodes is used for scalable linkage. This extra storage could be avoided with the cost of a linear scan of data when scaling the linkage. The cells are part of spatial partitioning of locations. For this, we use S2[1], which divides the Earth's surface into 31 levels of hierarchical cells, where, at the smallest granularity, the leaf cells cover $1cm^2$.

We deliberately form the mobility history tree via hierarchical temporal partitioning, and not via hierarchical spatial partitioning. This is because spatial partitioning is not effective in detecting negative linkage (alibi [21]). Recall, given two locations in the same temporal window, if it is not possible for an entity to move from one of these locations to the other within the width of the window, then these two records are considered as a negative link, i.e., alibi. To calculate the similarity score of an entity pair, we compare their records to those that are in close temporal proximity. Record pairs that are close in both time and space are awarded, whereas record pairs that are close in time but distant in space are penalized. Hence, we favor fast retrieval of records based on temporal information over based on spatial information.

[1] `http://s2geometry.io/`

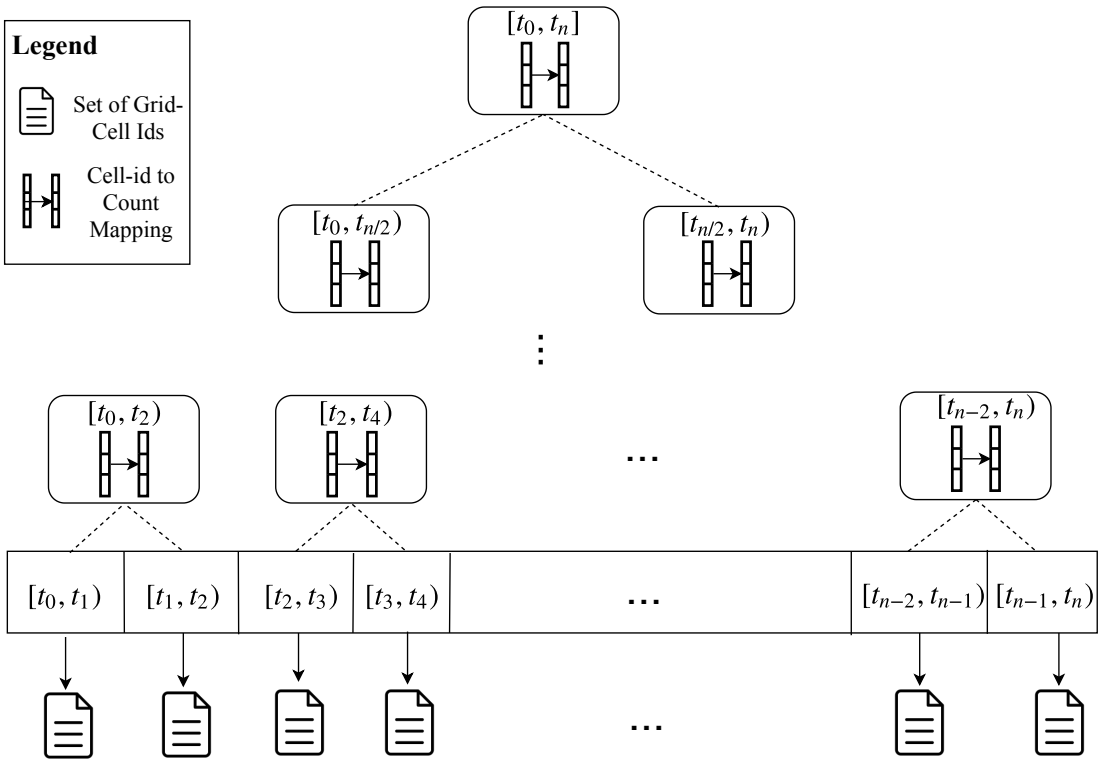


Figure 4.1: Mobility history representation

Both the temporal window size used for the leaf nodes and the S2 level (spatial granularity) used for the cells are configurable. As detailed later, we auto-tune the spatial granularity for a given temporal window size using the trade-off between accuracy and performance of the linkage.

Figure 4.1 shows an example mobility history of an entity. Each leaf keeps a set of locations, represented with cell ids. Each non-leaf node keeps the information on how many times a cell id has occurred at the leaf level in its sub-tree.

4.2 Overview of the Linkage Process

The linkage is performed in three steps. First, a Locality-Sensitive Hashing (LSH) based filtering step reduces the number of entity pairs that needs to be considered for linkage. It is an optional step that significantly improves scalability, yet has

minimal adverse effects on the precision and recall of the linkage.

The second step involves computing the pairwise similarity scores of entities from the two datasets. The similarity scores are computed based on a formula that determines how similar the mobility histories are. The computed similarities are used as weights to construct a bipartite graph of the entities from the two datasets.

The final step is to apply a maximum sum bipartite matching, where the matched entities are considered as linked.

4.2.1 Mobility History Similarity

The similarity score we use for comparing the mobility histories is inspired by the information retrieval literature, but has unique aspects taking advantage of the spatio-temporal nature of our domain.

In document matching, term weighting, cosine similarity, and tf-idf, as well as ranking techniques from the search domain, BM25 in particular, are adapted for inter-document similarity scores [33]. Inspired by these, we study the matching of mobility histories with an analogy by treating the lowest level of the mobility histories as documents where the spatial cell / time window pairs correspond to terms. We refer to these pairs as *time-location bin*s. Two location-based services are typically not synchronously used and thus time-location bins that are spatially close should contribute positively towards the similarity score. Therefore, we first define a proximity function to evaluate closeness of cells within the same time window. This proximity is a decreasing function of the geographical distance between the locations of the cells. The outcome becomes negative when it is not possible for an entity to move from one of these locations to the other within the width of the time window. These cell pairs are *alibi*s [21] — a concept unique to our domain.

The final step of calculating the similarity score of two mobility histories is to

sum the aforementioned proximities for all temporal windows. During this sum, similar to document similarity scores, we award the uniqueness of the matching time-location bins using an inverse time-location bin frequency term. Furthermore, the scores are inversely proportional with the ratio of the length of a mobility history to the average length of all histories from the same service. Finally, when calculating proximities, a cell is used only once. Thus, a matching algorithm is used to pair the cells from the same time window across two datasets. In particular, this algorithm takes two sets of cells and constructs a set of mutually nearest neighbor cell pairs. Once a cell is used to construct a pair, it is not used for further mutually nearest neighbor calculations to make sure that every cell is used exactly once.

4.2.2 Bipartite Matching

After the similarity scores are calculated, we construct a weighted bipartite graph and perform maximum sum bipartite matching where two ends of the selected edges are considered as matches. Once the matches are found, they are output as linkage only if their scores are above a threshold. We call this threshold the *linkage stop similarity threshold*. Based on our observation from test datasets, the score distribution often follows a binary Gaussian mixture distribution. As such, we decide on this cut point by matching a Gaussian Mixture Model (GMM) on the scores and finding the point that maximizes the $F1$-Score, assuming that the two Gaussians represent the false and true linkages, respectively.

4.2.3 Reducing the Number of Pairs

Considering that the mobile services have millions of entities interacting with them everyday, processing hundreds of days of data for matching would take impractically long time. In order to match entities in a reasonable time, we present a locality-sensitive hashing (LSH) based mechanism for mobility histories, which only calculates the similarity score of the mobility history pairs that fall

into the same hash bucket. The similarity score we use for the LSH is not the same score we have outlined for mobility histories, but is a simpler, cruder score that lends itself to LSH. As such, the LSH step slightly reduces the recall, but brings several orders of magnitude speedup in linkage.

4.3 Mobility History Linkage

In this section we describe the computation of similarity scores of mobility history pairs and the maximum weight bipartite matching for linkage. Since the set of entities present in the two location datasets are not necessarily identical and the amount of intersection between them is not known in advance, we also discuss how to find an appropriate score threshold to limit the number of entities linked.

4.3.1 Mobility History Similarity Score

The similarity score of a pair of mobility histories should capture closeness in time and location. This score should not require a consistent matching of time-location bins across two histories. This is because the mobility histories are not synchronous, i.e., a record is not necessarily present for the same timestamp in both of the datasets. Therefore, the similarity score needs to aggregate the proximities of the time-location bins, different from the traditional techniques where similarity measures are defined over records, like Minkowski distance or Jaccard similarity. We define a number of desired properties for the similarity score:

1) Award the matching of close time-location bins. While the similarity score contribution of exactly matching time-location bins is obvious, one should also consider bins that are from the same temporal window but are from different yet spatially close cells. Such time-location bins are deemed close and they should contribute to the similarity score relative to their closeness.

2) Tolerate temporal asynchrony, that is, do not penalize the similarity when one entity has records in a particular time window but the another does not. This case is common when the location datasets are obtained from asynchronous services. While spatio-temporally close usage should contribute to the similarity, lack of it for a particular time window should not penalize the score.

3) Penalize the matching of alibi time-location bins. Time-location bins that are from the same time window but whose cells are so distant in space that it is not possible for an entity to move between these cells within the time window are considered as alibi bins. They should be considered as counter evidence in terms of linking the entities and penalized in the similarity score.

4) Award infrequent cells in matching time-location bins. While summing up the proximities of the matched time-location bins, the uniqueness of the cells should be awarded as they are stronger indicators of similarity than cells that are seen frequently.

5) Normalize the similarity score contributions based on the size of the mobility histories in terms of the number of time-location bins. If not handled properly, the skew in the number of time-location bins would result in the mobility histories with too many bins to dominate the similarity scores over the shorter ones.

In what follows, we first define the proximity of time-location bins and then present our algorithm for aggregating these to compute the final score.

4.3.1.1 Proximity of Time-Location Bins

One of the desired properties of the time-location bin proximity was tolerating temporal asynchrony. Therefore, we consider only the temporally close time-location bins in our proximity computation.

If two time-location bins are associated with the same temporal window, one could say that they are temporally close. Let $\mathcal{T}$ be a binary function that takes the value of 1 if the two given time-location bins satisfy this property and 0

otherwise. Let $e = \{w, c\}$ and $i = \{w, c\}$ be two time-location bins where $e.w$ and $i.w$ are two temporal windows and $e.c$ and $i.c$ are two spatial grid cells. We define $\mathcal{T}$ formally as $\mathcal{T}(e, i) = [e.w = i.w]$.

We define the spatial proximity of a pair of time-location bins as an inverse function of the geographical distance of their locations. However, we use an upper bound on the geographical distance to capture the concept of alibis. This upper bound, referred to as the *runaway distance*, is defined as the maximum distance an entity can travel within the given temporal window. It is represented as $\mathcal{R}$ and calculated by multiplying the width of the temporal window with the maximum speed, α, at which an entity can travel. Let w be a temporal window and $|w|$ be the width of it, then $\mathcal{R} = |w| \cdot \alpha$. We define the proximity function $\mathcal{P}$ formally as follows:

$$\mathcal{P}(e, i) = \mathcal{T}(e, i) \cdot \log_2 \left(2 - min(d(e.c, i.c)/\mathcal{R}, 2)\right) \tag{4.1}$$

where d is a function that calculates the minimum geographical distance between two grid cells.

When a given pair of time-location bins are not from the same temporal window, the outcome of this function becomes 0. When they are from the same temporal window and the same spatial grid cell, the outcome becomes 1 — the highest value it can take. As the distance increases up to the runaway R, the value goes down to 0 with an increasing slope. If the distance is more than R, the outcome becomes negative, reaching $-\infty$ as the distance reaches two times the runaway distance. This is a simple yet effective technique to capture the alibi record pairs. In a real-world setting, the data might have inaccurate information regarding the location of an entity. Therefore, while the decrease to negative values is steep, it is still a continuous function that allows a small number of alibi record pairs whose distance from each other is slightly larger than the runaway distance.

The proximity function $\mathcal{P}$ is designed so that our similarity score satisfies the first three of the required properties we have outlined earlier, namely: awarding the matching of close time-location bins, tolerating temporal asynchrony, and penalizing the alibis.

4.3.1.2 Aggregation of Proximities

The similarity score computation is performed over the time-location bins at the leaves of the mobility histories. For entity $u \in U_{\mathcal{E}}$ (and $v \in U_{\mathcal{I}}$), we represent the set of time-locations bins as H_u (and H_v), where $e \in H_u = \{c, w\}$ (and $i \in H_v = \{c, w\}$) represents a time-location bin with the time window w and the grid cell c. Recall that in a mobility history, a set of grid cells are kept for each time window. These are the grid cells that contain at least one record for the entity, during the given time window.

Given a pair of mobility histories, the computation starts with constructing pairs of time-location bins whose proximity will be computed and included in the aggregation. At one extreme, one could perform a Cartesian product over the mobility histories and include all resulting pairs in the aggregation. However, this would be over-counting, as a time-location bin will end up participating in multiple pairs. At another extreme, one could use pairs that have the exact same time-location bins. Obviously, this would violate the first desired property by ignoring the close ones. Therefore, we first introduce a pairing function $\mathcal{N}(u, v)$, which computes the set of time-location bin pairs to be included in the aggregation for a pair (u, v) of mobility histories.

We restrict ourselves to time-location bin pairs from corresponding time windows, as this guarantees that the pairs satisfy the temporal proximity, $\mathcal{T}$. As such, we have $\mathcal{N}(u, v) = \bigcup_w \mathcal{N}_w(u, v)$. Given a time window w, we compute $\mathcal{N}_w(u, v)$ by first selecting the time-location bin pair (e, i) with $e.w = i.w = w$ that has the smallest geographical distance $d(e.c, i.c)$. This pair also satisfies the property that the time-location bins are *mutually nearest neighbors*. Once this pair is added into $\mathcal{N}_w(u, v)$, pairs containing any one of the selected time-location bins, that is e or i, are removed from the remaining set of candidate pairs. The addition of the mutually nearest neighbor pair into $\mathcal{N}_w(u, v)$ and the filtering of the candidate pairs continue iteratively until there are no time-location bins left in the smaller mobility history.

Once the set of time-location bin pairs to include in the aggregation are found,

we define the similarity score function of two mobility histories for the entities $u \in U_{\mathcal{E}}$ and $v \in U_{\mathcal{I}}$, as follows:

$$\begin{aligned} \mathcal{S}(u,v) = & \sum_{\{e,i\} \in \mathcal{N}(u,v)} \mathcal{P}(e,i) \, \frac{\min\left(idf(e,\mathcal{E}), idf(i,\mathcal{I})\right)}{\mathcal{L}(u,\mathcal{E})\mathcal{L}(v,\mathcal{I})} \\ \text{where:} \quad & \mathcal{L}(u,\mathcal{E}) = (1-b) + b \frac{|H_u|}{\sum_{u' \in U_{\mathcal{E}}} |H_{u'}| / |U_{\mathcal{E}}|} \end{aligned} \tag{4.2}$$

The similarity function, $\mathcal{S}$, has three main components. The first component is the proximity $\mathcal{P}$, as defined in Eq. 4.1. For the time-location bin pairs identified by the pairing function $\mathcal{N}$, we sum up the proximities. The other two components deal with the scaling of the proximity value, in order to *i*) normalize the differences in the mobility history sizes in terms of the bin counts, and *ii*) award infrequent cells in the matching. These two properties implement the last two desired properties from Sec. 4.3.1.

When one of the histories has more time-location bins, compared to the average size of other histories from the same location dataset, it is more likely for it to have matching records. This would cause the histories with the most number of bins being highly similar with all other histories from the opposite dataset. Therefore, we introduce a normalization function $\mathcal{L}$, for both histories, which makes the contribution of each bin pair to the similarity score inversely proportional with the relative sizes of the histories. The relative size of a mobility history is defined as the ratio of the number of time-location bins it contains over the average number of time-location bins from the same dataset.

In order to tune the impact of the mobility history size in terms of time-location bins, we add a parameter b, which takes a value between 0 and 1. At one extreme, $b = 0$, the denominator becomes 1, i.e., the history lengths are ignored. At the other end, $b = 1$, the denominator becomes the product of the relative history sizes. We have borrowed this technique from BM25 [34] — a ranking function used in document retrieval, which uses this technique to avoid bias towards long documents.

The final component of the similarity score function is the *idf* multiplier. This

component is analogous to inverse document frequency from the document clustering domain. The idea behind it is to award uniqueness of a pair of time-location bins. If an entity is in an infrequent time-location bin, i.e., the number of other entities from the same dataset in the same time-location bin is low, the contribution of this bin to the similarity score should be higher. Likewise, if a time-location bin is highly common among entities of one dataset, the contribution should be lower. As the frequency of time-location bins might differ across datasets, we take the *idf* score that makes the lowest contribution. *idf* of a time-location bin equals to the logarithm of the ratio of the number of mobility histories to the number of mobility histories that contain the given time location bin. Formally, given a time location bin, $e \in H_u$, for the entity $u \in U_{\mathcal{E}}$, we calculate $idf(e, \mathcal{E})$ as follows:

$$idf(e, \mathcal{E}) = \log\left(|U_{\mathcal{E}}| / |\{u \in U_{\mathcal{E}} \mid e \in H_u\}|\right) \tag{4.3}$$

4.3.2 Mobility History Linkage via SLIM

To effectively realize the linkage of mobility histories using our similarity score, we have designed the *SLIM* algorithm, which is shown in Algorithm 2. *SLIM* starts by creating the mobility histories from the two given location datasets. It then finds the mutually nearest neighbour pairs for each corresponding time window. The similarity score is computed by aggregating the weighted proximities for these pairs, as it was outlined in Eq. 4.2.

The *SLIM* algorithm takes an additional step at increasing the effectiveness of alibi detection. In a given time window, while the distance between mutually nearest pairs may not be greater than the runaway distance, it might be possible for the mutually furthest pairs to have this property. To capture alibi time-location pairs better, we also compute the set of mutually furthest pairs, and add their proximity to similarity score as well, if an alibi is detected. The FN (mutually furthest neighbor) function in the algorithm acts similar to the MN (mutually nearest neighbor) function, but it chooses the pairs with the furthest

[2]We overload the notation such that $\mathcal{S}$ takes the bin pairs as input.

Algorithm 2: SLIM: Scalable Linkage of Mobility Histories

Data: $\mathcal{E}, \mathcal{I}$: Location datasets to be linked
Result: $\mathcal{L}$: Linked pairs of entities

```
H_E ← CreateHistories(E) ;            // Histories from first dataset
H_I ← CreateHistories(I) ;            // Histories from second dataset
E ← {} ;                                          // Initialize edges
V ← {} ;                                        // Initialize vertices
for H_u ∈ H_E do                  // For each history in first history set
    W_Hu ← H_u.getAllWindows() ;                // Get all windows of H_u
    for H_v ∈ LSHFilterPairs(u) do             // For a candidate history
        S ← 0 ;                                // Initialize similarity score
        W_Hv ← H_v.getAllWindows() ;            // Get all windows of H_v
        for w ∈ W_Hu ∩ W_Hv do                        // For a common window
            N ← MN(u, v; w) ;                       // Mutually nearest pairs
            S ← S + S(N) ;              // Update similarity (see Eq. 4.2²)
            N' ← FN(u, v; w) ;                    // Mutually furthest pairs
            for (e, i) ∈ N' do             // For each mutually furthest pair
                D ← S({(e, i)}) ;                           // Delta similarity
                if D < 0 then                       // If an alibi is detected
                    S ← S + D ;                            // Update similarity
        if S > 0 then                                  // If score is positive
            V ← V ∪ {u, v} ;                             // Add to vertex set
            E ← E ∪ {(u, v; S)} ;                 // Add to weighted edge set
L ← LinkPairs(G(E, V)) ;                             // Find linked entities
return L ;                                  // Return linked entity pairs
```

distance, instead of the closest. To avoid double counting, we *only* consider these pairs if they are alibis.

Once the similarity scores are computed for the mobility history pairs, they are used to construct a weighted bipartite graph. If the score is negative, no edges are added to the graph. Next, we describe how to perform maximum sum bipartite matching and decide a stop point for the linkage.

4.3.3 Maximum Weighted Bipartite Matching

The *SLIM* algorithm computes a weighted bipartite graph $G(E, V)$, where $V = U_{\mathcal{E}} \cap U_{\mathcal{I}}$ and E is the set of edges between them. This bipartite graph is used to find a maximum weighted matching where the two ends of the selected edges are considered linked. To avoid ambiguity in the results, the selected edges should not share a vertex. Since the matching is performed on a bipartite graph, there is no edge between two entities from the same dataset.

This matching problem is one of the fundamental combinatorial optimization problems known as *assignment problem*. There are many solutions to this problem, one of which is the Hungarian Algorithm [35]. This algorithm provides an optimal solution in polynomial time, with a worst-case complexity of $\mathcal{O}(n^3)$, where $n = max(|U_{\mathcal{E}}|, |U_{\mathcal{I}}|)$. There are also many approximation algorithms available in the literature [36]. In this work, instead of putting our effort on a new matching algorithm, we use a simple greedy heuristic, which links the pair with the highest similarity at each step. We focus on the problem of finding a stopping condition for the linkage as part of the bipartite matching process. This is because our experimentation has shown that the stopping condition has a pronounced impact on precision and recall, yet the use of a heuristic vs. exact solution for the bipartite matching does not make a noticeable impact on accuracy.

Maximum weighted matching algorithms, including the Hungarian algorithm, are designed to find a full matching. In other words, all entities from the smaller set of entities will be linked to an entity from the larger set. However, in a real-world setting for linking mobility histories, it is unlikely to have the entities from one dataset to be a subset of the other. Frequently, it is not even possible to know the intersection amount in advance. This is an important but so far an overseen issue in the related literature [14, 19]. Therefore, we implement a mechanism to decide an appropriate threshold score to stop the linkage, to avoid false positive links when the sets of entities from two location datasets are not identical.

After performing a full matching over the bipartite graph, there are two sets of selected edges. The first set is the true positive links, which contains the selected

edges whose entities at the two ends refer to the same entity in real life. The second set is the false positive links, which contains the rest of the selected edges. The purpose of defining a score threshold for stopping the linkage is increasing the precision, by ruling out some of the selected edges from the result set. Ideally, this should be done without harming the recall, by extracting edges only from the false positive set. However, this is a challenging task when ground truth or training data does not exist.

A good similarity score should allow to group true positive links and false positive links in two clusters. Moreover, these two clusters should be distinguishable from each other. With the assumption that our similarity score satisfies these properties, to determine the stop threshold, we first fit a 1-dimensional Gaussian Mixture Model (GMM) with two components over the distribution of the selected edge weights [37]. We assume that the two components have independent Gaussian distribution of weights and the component with the larger mean models the true positive links (the other modelling the false positive links). Let us call the component with the smaller mean m_1 and the component with the larger mean m_2. Assume that there is already a similarity score threshold s. The cumulative distribution functions of the components m_1 and m_2 are used to compute: i) the area under the curve of m_1 and to the right of $y = s$ line, which gives the rate of false positives, and ii) the area under the curve of m_2 and and to the right of $y = s$ line, which gives the rate of true positives. Using these two rates, we calculate the expected precision as shown in Eq. 2.1.

Using precision and recall, one could calculate a combined F1-score as $F1(s) = 2P(s)R(s)/(P(s) + R(s))$. Note that all these scores are dependent on the score threshold s. Let c_1 ad c_2 denote the weights of the GMM components m_1 and m_2, respectively. We have $R(s) = c_2(1 - F_{m_2}(s))$ and $P(s) = R(s)/(R(s) + c_1(1 - F_{m_1}(s)))$, where F represents the cumulative distribution function. Finally, we have $s^* = argmin_s F1(s)$ as the linkage stop score threshold to use.

We only include the links whose edges are higher than the threshold in the result. Figure 4.2 shows an example of GMM fitting on sample similarity scores.

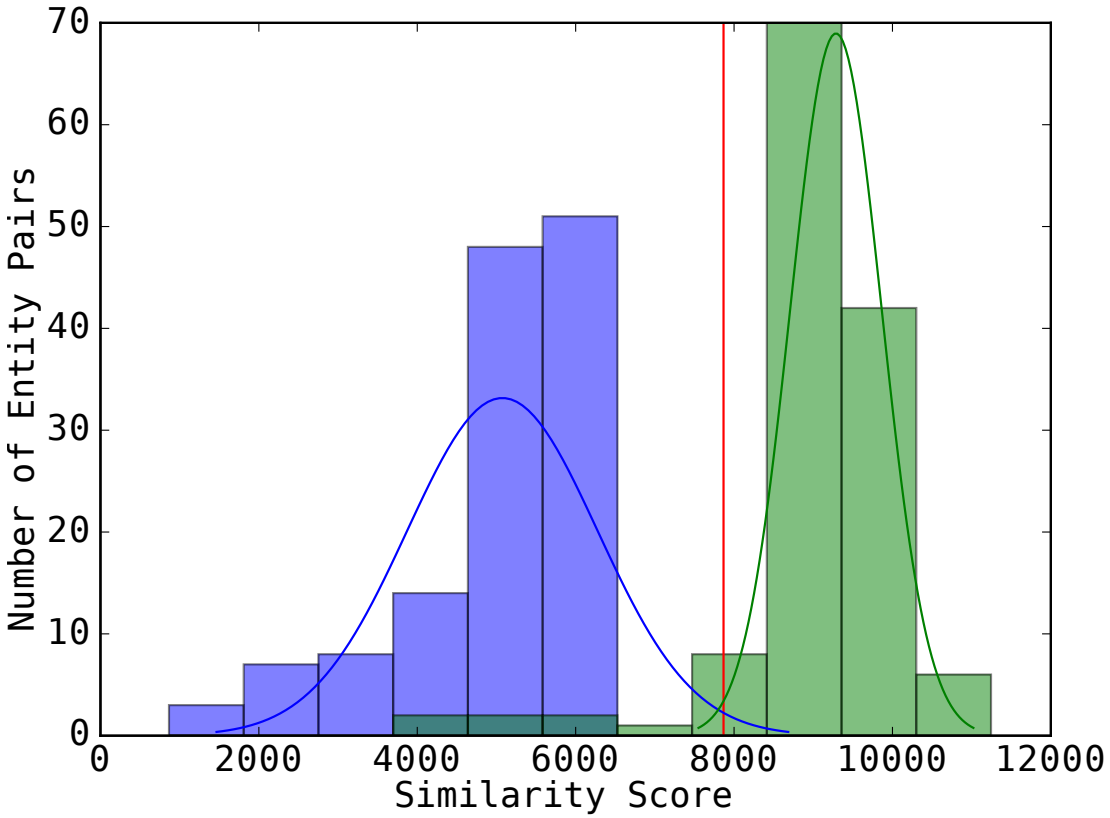


Figure 4.2: Sample GMM fit for similarity scores

The x-axis shows the scores and the y-axis shows the number of links in a particular score bin. The two lines show the components of the GMM. The green bars in the histogram show the number of true positive links and the blue bars show the number of false positive links (obtained from the ground truth and shown for illustrative purposes). Vertical red line is the detected linkage stop similarity score threshold value. For the detection of the threshold, we have also experimented with different techniques, namely 2-means clustering and the Otsu method [38]. Our experimentation showed that the proposed GMM-based technique performs better than these alternatives in providing a good balance of precision and recall.

4.3.4 Parameter Tuning

One subtle issue is identifying width of the temporal window, and the spatial level of detail to use. Existing work either define them using previously labeled

data [19] or use preset values identified via human intuition [14]. Here, we take a step forward to automatically tune the spatial level for a given temporal window width, in the absence of previously labeled data.

The trade-off when deciding the spatial level is between accuracy and performance. When the spatial domain is coarsely partitioned, the record locations become indistinguishable. At one extreme, the entire globe is a single grid cell and all events are on the Earth. On the other hand, increasing the spatial detail, hence creating finer grained partitions of space, increases the size of the mobility histories. Linking larger histories takes more processing time. At another extreme, similarity score computation of a pair of histories with a single window and with maximum spatial detail leads to pairwise comparisons of all records. On the other hand, increasing the spatial detail does not always improve the accuracy of the linkage. Our observations based on experiments with multiple datasets show that, after a certain level of detail, increasing the spatial detail neither improves nor worsens the accuracy of the linkage.

To find out the best spatial grid level that balances the aforementioned trade-off, we perform a test on distinguishing entities from the same dataset. When the level of detail is too low, similarity scores of all pairs would be close to each other. In this case, the similarity score of entity pairs u and v will be close to the similarity score of u and u (self-similarity). Using higher details of spatial information would decrease this ratio, indicating an entity is more similar to itself than any other entity. Making use of this observation, we first select a subset of entities from the dataset and form a set of pair of entities by crossing them with the rest of the entities. Next, for changing spatial level of detail, we compute the average of the aforementioned ratio (pair similarity over self-similarity) for all pairs. Once we have the average values for each spatial detail, we perform a best trade-off point detection algorithm (aka. elbow point detection) as implemented in [39]. Repeating this procedure independently for two datasets to be linked, we use the higher knee point as the spatial detail level of the linkage. In our experimental evaluation we show that this technique is able to detect most accurate spatial detail level that does not add overhead in the performance, for a given temporal window.

4.4 Locality Sensitive Hashing for Scalability

Comparing each entity pair would require $\mathcal{O}(N \cdot M)$ similarity score computations, where $N = |U_{\mathcal{E}}|$ and $M = |U_{\mathcal{I}}|$. Considering large number of entities and even larger collection of records being produced, *Locality-Sensitive Hashing* (LSH) [23] can be utilized to focus on pairs that are likely to be similar.

LSH is based on a family of hash functions $\mathcal{H}$, mapping similar items to the same hash code with higher probability than dissimilar items. One of the most popular of these families is min-hash [40]. An application of LSH with min-hash for document matching works in three steps. First, each document is represented as a set of *k-shingles*: set of characters (or words) of length k. The similarity between documents is defined as the Jaccard similarity of this set of shingles. Second, these sets are converted to short signatures using min-hashing. This procedure is defined as follows: Pick a random permutation of the ground universe of the shingles, and for each document append the smallest shingle rank in the document to its signature. The length of the signature is defined by the number of times this procedure is repeated. The probability that the min-hash function produces the same value for two sets is equal to the Jaccard similarity of them [41]. This property guarantees that the similarity of sets is preserved in similarity of signatures. In the final step, these signatures are hashed to a large number of buckets multiple times using a technique known as *banding*. In the banding step, signatures are divided into b bands consisting of r rows (signature size equals to $b \cdot r$). Then, the same hash function is used to hash all bands, but for each band a separate bucket array is kept. This prevents identical bands in different parts of the two signatures to cause them to be hashed into the same bucket. Intuitively, the more similar the signatures are the more likely that they have at least one identical band [41]. If two documents are hashed to the same bucket for at least one band, they are considered as *candidate pairs* and further processing is applied to them.

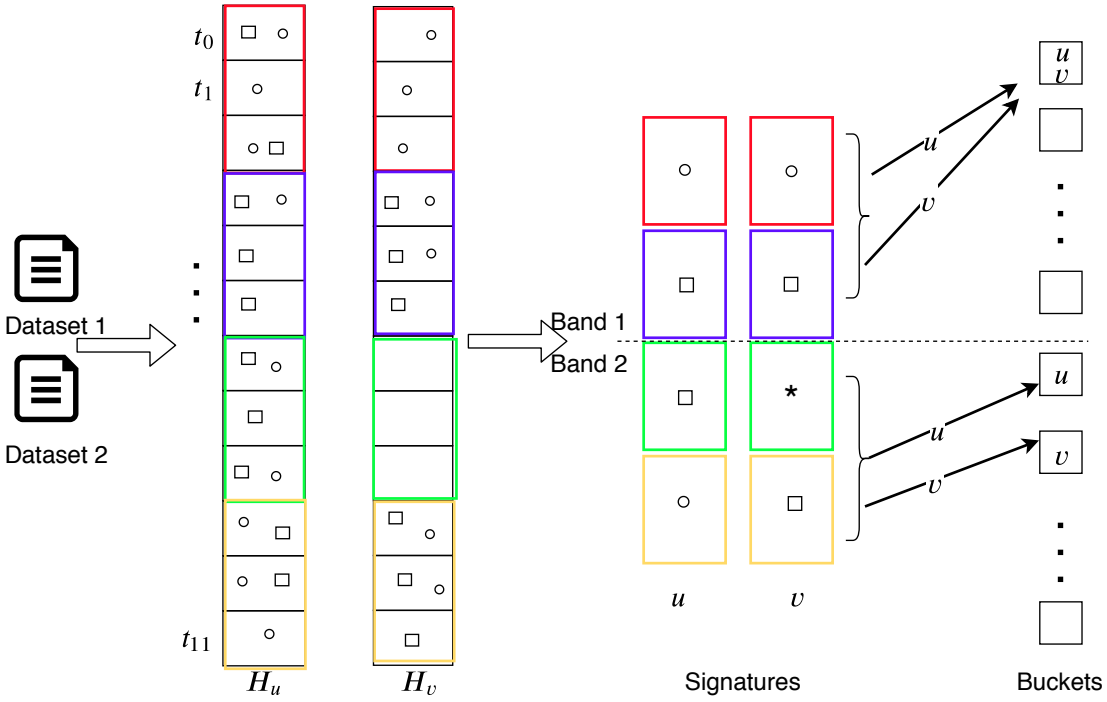


Figure 4.3: Locality-sensitive hashing of mobility histories

4.4.1 LSH for Mobility Histories

In a real-world setting, location datasets are generally asynchronous and sparse. Under such conditions, representing the mobility histories as sets of *k-shingles* of records and expecting identical bands when applying the banding technique would be overly strict. On the other hand, for two entities to link it is expected that most of their records are generated in the same spatial grid cell. Such cells are known as *dominating grid cells* [21], which contain most event records of the owner entity and are determined by simply counting the entity's records for different cells and picking the cell with the highest count. While one dominating grid cell could be found using the entire dataset, it is also possible to specify a start and end time to form a query, and find the dominating grid cell for a particular time window defined by this query.

Given a mobility history, we construct a list of dominating grid cells to act as signatures. This construction is done by querying each mobility history for non-overlapping time windows to find the corresponding dominating grid cells,

and adding the resulting cells to the end of the signature. We make sure that the queries span the same time period with the data, and order of the queries is the same across mobility histories. If a mobility history does not contain any record for a query time window, a unique placeholder is added to the signature to make sure signatures have the same structure. In other words, the dominating grid cells obtained from the same index of different signatures are results of the same query. Later, when applying hashing, these placeholders are omitted.

In summary, each mobility history is converted into a signature consisting of a sequence of dominating grid cells. The similarity t across two signatures is defined as the number of matching dominating grid cells, divided by the signature size. The signature size is determined by the query window size. Query window size, which is a parameter of our LSH procedure, is a multiple of the leaf-level window size. The appropriate level of the mobility history tree is used to quickly locate the dominating grid cells.

With the mobility history signatures at hand, we next apply the banding technique, like in the case of document matching. Recall, in the banding technique the signatures are divided into b bands consisting of r rows, and each band is hashed to a large number of buckets. The goal here is to come up with a setting such that signatures with similarity higher than a predefined threshold t to be hashed to the same bucket at least once.

Given two signatures of similarity t, the probability of these signatures having at least one identical band is $1 - (1 - t^r)^b$. Regardless of the constants b and r, this function has an S-Curve shape and the threshold t is the point where the rise is the steepest. Consequently, it is possible to approximate the threshold t as $(\frac{1}{b})^{\frac{1}{r}}$ [41].

It is possible to calculate the number of bands, b, to reach a particular threshold, t. Let s be the signature size and t be the similarity threshold for becoming a candidate pair, the number of bands, b is calculated as follows. Given $t = (1/b)^{1/r}$ and $r = s/b$, we have $t = (1/b)^{b/s}$. Solving for b gives:

$$b = e^{\mathcal{W}(-s \ln t)} \tag{4.4}$$

where $\mathcal{W}$ is the *Lambert W* function, which is the inverse of the function $f(x) = x\,e^x$ [42]. The parameters to the LSH procedure for mobility histories are: i) the similarity threshold t, ii) the query time window size for determining dominating grid cells, and iii) the spatial level of the dominating grid cells. We explore the setting of these parameters as part of the experimental evaluation.

Since the similarity between the signatures is Jaccard similarity and the banding technique applied as it is in the LSH with MinHash [41], the probabilistic guarantees that are provided by LSH are preserved. The only difference between the introduced LSH and the previous application is instead of forming the signatures from documents using random permutation of *k-shingles* we generate them querying mobility histories multiple times. The relationship between the similarity of mobility histories and the similarity of signatures is intuitive.

4.4.2 Illustrative Example

Given a collection of records of entities, they are first converted into mobility histories. Leaf levels of two mobility histories for the entities u and v , are shown in the figure 4.3 (H_u, H_v). In this representation each history consists of 12 time windows. For demonstration purposes, we assume that entities visit only two spatial grid cells, represented with *square* and *circle*. Note that there are no duplicate time-location bins.

While it is not shown in the figure, before performing the linkage of two datasets we first link each dataset to itself to identify an ideal spatial level of detail that balances the tradeoff between performance and accuracy as explained in section 4.3.4. Since this self linkage uses the labeled data, it is possible to find the lowest level of spatial detail that helps to better distinguish between an entity's self similarity with it's similarity to other entities. Intuitively and as it will be demonstrated in our experiments, after a certain level, increasing the detail further would not effect the ratio of pair similarity over self similarity but would harm the performance.

Once the spatial level of detail is identified, we next apply LSH. Given two mobility histories, we first query the mobility histories four times for finding the dominating grid cells. Each query has a window size of three time units and is shown in different colors. The resulting signatures are of length 4, For the first query (red rectangle), entity u visited the grid-cell *circle* 3 times and it visited the grid cell *square* 2 times. Therefore, the dominating grid cell for entity u during the first query time window is grid-cell *circle*. This procedure is repeated for all queries and entities. The third index of the signature for entity v has the mark $*$ because it has no records during the third query's time window (green rectangle). Once the signatures are ready, we apply the banding technique using two bands. For the first band, since the signatures are identical, the entities are hashed to the same bucket. In this example setting, the threshold for becoming a candidate pair is $t = 0.5^{\frac{1}{2}} \approx 0.7$ and the similarity of signatures is 0.75.

Since the entities u and v are candidate pairs, in the next step we compute the similarity score of this pair. In the first time window, upper most cell, entity u has visited two distinct grid cells, *circle* and *square*, and entity v visited *circle* grid cell only. The pairing function $\mathcal{N}$ will take these two sets of locations and will compute the set of record pairs to be included in the aggregation. In this example this function will pair *circles*, as the distance between them are minimum (i.e. mutually nearest neighbours). The contribution of this pair to the similarity score will be computed according to equation 4.2 that takes proximity of the grid cells, their popularity and length of mobility histories into account. Lastly, the algorithm would check if there is an alibi pair of events in the given set. This time, we use $\mathcal{N}'$ as the pairing function and it will pair *square* and *circle* grid cells as they are mutually furthest neighbours of these two sets of records. Depending on the geographical proximity of these two cells, this pair might have a negative contribution to the aggregation (alibi). This procedure will be repeated independently for each temporal window and computed scores for each time window will be summed. Once the computation is finished the resulting score is be used to set weight of the edge between entities u and v in the final bipartite graph.

Last step of our algorithm is to perform the maximum weighted bipartite matching and assume select edges as our links. Once these edges are identified,

to avoid false positive pairs, we fit a 1-dimensional GMM with two components over the weights of these edges and determine the stop threshold. We report only the edges that has higher weight than this determined stop threshold. In our experimental setting, LSH brings two to four orders of magnitude speedup to linkage with a slight reduction in the recall. To the best of our knowledge, this is the first locality-sensitive hashing based approach for mobility history linkage. Moreover, similarity score definition and stop threshold determination algorithms are proven to be accurate as they reach more than 0.9 $F1-$score in all settings.

Complexity Analysis. The generation of mobility histories takes $\mathcal{O}(N + M)$ time. where $N = |U_{\mathcal{E}}|$ and $M = |U_{\mathcal{I}}|$. Since we form the mobility histories in the form of a tree, cost of generation of the LSH signatures is reduced to linear time with signature size instead of linear time number of records. The banding step of the LSH has the worst case time complexity of $\mathcal{O}(N + M)$. This is the case when each record itself is an entry in the signature. After LSH is completed, the linkage takes $\mathcal{O}(N \cdot M)$ time if all records are in the same temporal window and from distinct spatial cells and LSH cannot reduce any entity pairs. In the last step, we use a simple greedy heuristic that sorts $|U_{\mathcal{E}}| \cdot |U_{\mathcal{I}}|$ links by their weights and has $\mathcal{O}(|U_{\mathcal{E}}| \cdot |U_{\mathcal{I}}| \cdot log_{|U_{\mathcal{E}}| \cdot |U_{\mathcal{I}}|})$ time complexity. While the worst-case complexity of linkage is quadratic, our experimental evaluation shows that running time of *SLIM* algorithm is linear with the data size.

Chapter 5

Beyond Linkage: Extensions

In this section, we extend our solutions to handle spatial uncertainties and dynamic changes in data. Moreover, we discuss how to perform linkage when there are more than two datasets to merge.

5.1 Handling Spatial Uncertainties

In a real world setting, the location information contained in records might be in the form of a region, or uncertain. This might be a result of imperfect data collection or data including a reference to a region instead of entity's exact location.

For the proposed rule based approach the location information contained in records are in the form of regions by default. Yet, it can handle the point locations by looking at the distance between two points being below a maximum distance threshold, instead of looking for region intersection. On the other hand, for the similarity based approach, if a record includes a reference to a region instead of a point, it could be copied to multiple time-location bins of the mobility history, as such the union of locations of these bins covers at least the record region. At one extreme, using too many of too small grid cells, one could cover the exact region but would create many time-location bins. The storage overhead caused by this

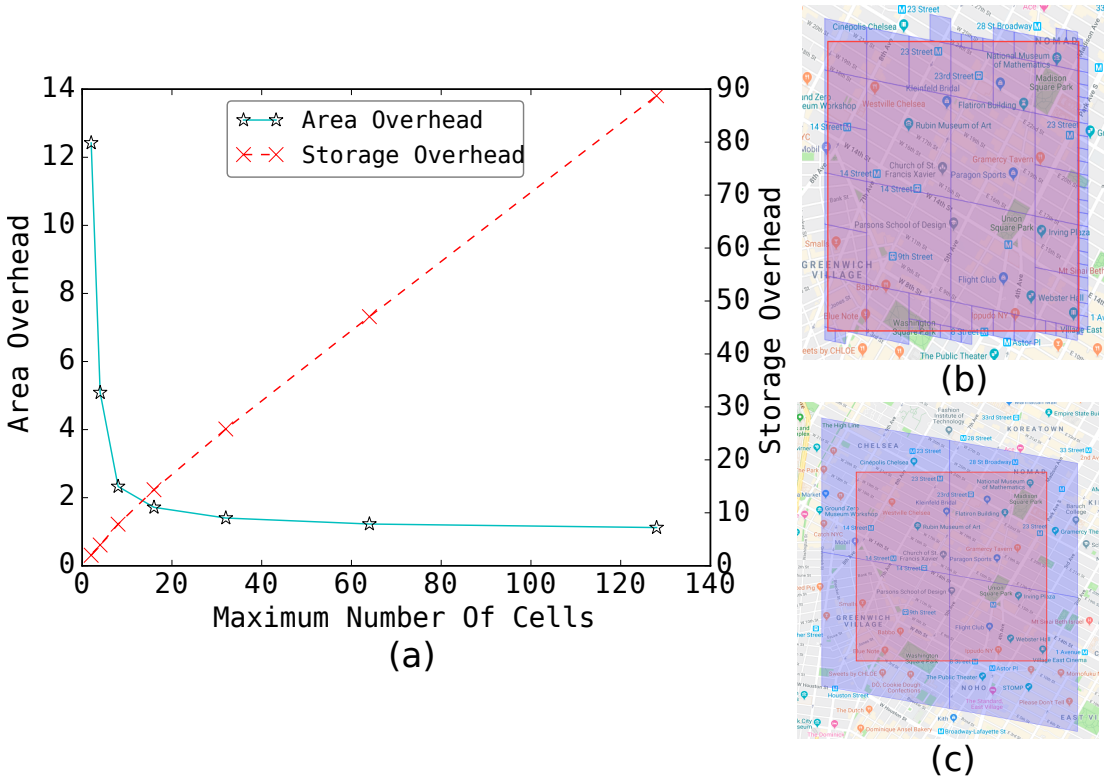


Figure 5.1: Handling Spatial Uncertainty

copying could be reduced by decreasing spatial level of detail of grid cells, with a sacrifice on location accuracy. At another extreme, all regions are on the Earth, a single time-location bin could cover the input region without introducing any storage overhead. Figure 5.1b and c illustrate this approach. In the first one, the region, red rectangle, is represented by union of 73 smaller grid cells and the area covered by this union almost as the same as the region of the initial cell. This coverage requires copying the record 73 times but the spatial information is as accurate as spatial information given in the record. In the latter, we use larger grid cells to represent the same region, using only 4 of them, but the area covered by the union is almost twice of that of input region. In this case storage overhead is decreased 24 times, by sacrificing spatial accuracy by covering 2 times of the input area. To better illustrate the trade-off between area overhead and storage overhead, we repeat the same test for $250,000$ records those have location information in the form of rectangles. For each rectangle, we limit number of cells to be used to cover the region and compute average number of cells to represent

a given region (storage overhead), and ratio of sum of areas of those cells to the area of the input region (area overhead). Figure 5.1a shows this experiment where x-axis is the maximum number of cells and left y-axis is area overhead and right y-axis is storage overhead. Dotted red line belongs to right y-axis and solid blue line belongs to the left y-axis. In this figure, we observe, it is possible to cover a region only with 12% extra area coverage if the data is replicated to 15 other cells. Moreover, in our dataset copying a record to more than 16 cells does not contribute to spatial accuracy. As this experiment illustrates, in a real world setting where location information of records are in the form of regions, the record should be copied to multiple cells when forming the mobility histories. The overhead caused by this process could be limited with a small sacrifice from the spatial accuracy.

5.2 Dynamic Changes in Data

Dynamic changes in data, e.g., in the form of event record appends, are common in mobility datasets. Our solutions can address such dynamic changes by avoiding performing the complete pipeline of the linkage. In the current structure, both counting the co-occurring events for k-l diversity and the similarity score computation are cumulative. Therefore, the arrival of an update will only trigger updating the k-l diversity and similarity scores.

In the rule based linkage, when a new record arrives, one should update the k-l diversity scores of all candidate entity pairs, whose one end is the record owner. There might be three effects of this update based on the initial state of the k-l diversity scores. First, if the record owner is not matched with any other entity in the previous computation (i.e. does not have a pair entity that satisfy k-l diversity) this update might introduce a new match. Second, if the record owner already has a matching entity but incoming record creates ambiguity in the linkage (i.e. there are more than one pairs of the owner entity those satisfy k-l diversity) the previously matched pair should be withdrawn from the result. Lastly, if the record owner already has a matching entity but the incoming record

does not create ambiguity, only k-l diversity scores should be updated.

One could optimize the handling of incremental updates for the rule based approach in two different ways. First, instead of applying updates for each record, it is possible to group the arriving records by the owner entity and apply them in batches. This will decrease the number of event retrievals from disk and number of updates required to process all new records. Second, since the contribution of each co-occurring record pair to k-l diversity scores is limited, instead of checking for all candidate pairs, one could avoid candidates those are unlikely to create a new link or ambiguity.

Similarly, in the similarity based approach, a new record should only effect the similarity scores of the record owner and the entities that are considered similar to it by the locality sensitive hashing. One subtle issue here is that, instead of updating the scores with every new record, one should update the scores for every mini-batch of records. These mini-batches should contain records from a certain time-window that has the same temporal width as the time window of mobility histories. Hence, it would be possible to apply time-location bin aggregation and decrease number of updates.

Unlike the rule based approach, the effect of dynamic changes in data on linkage might not be limited to only updated similarity scores. This is because maximum weight bipartite matching used in the final linkage is limited to create one to one links and one updated score might effect multiple decisions. Yet, performing final linkage periodically (i.e,. after a certain period of time or certain number of updates) is sufficient enough to keep the linkage up-to-date.

5.3 Extending Linkage to Multiple Datasets

The linkage process is much more complex when there are more than 2 datasets to unify. While this problem is orthogonal to ours and out of scope of this work, we still take an initial step forward with the following discussion.

Fundamental requirement of extending linkage to multiple datasets is making the similarity score computation, matching, and merging steps independent from the number of datasets. Therefore, it would be fair to assume that there are black box link and merge functions to be used to unify datasets. In our specific case these black-box link and merge functions have two inputs, datasets to be linked, and outputs a single unified dataset in which records of the linked entities are grouped together. The link function is the complete process explained in this paper and merge function is simply the union of the datasets.

One approach of multi dataset linkage is via separate pairwise matching of the datasets. But, this approach would result in poor scalability and would be lack of transitivity of decisions and thus require resolving discrepancies [43]. In such an approach linking k datasets would take $\frac{k\times(k-1)}{2}$ separate pairwise linkages [44]. On the other hand, existing research shows that if the aforementioned link and merge functions satisfy certain properties, linkage of all dataset pairs could be avoided [24] without sacrificing accuracy of the linkage. These properties are idempotence, commutativity, associativity and representativity (ICAR) and they should be satisfied in the record level and the dataset level. In this case, it is possible to find an order to link the datasets (or create a join tree) that would give the same accuracy with the brute force linkage and the time complexity would be linear with the number of datasets. In the database literature, this is akin to well known join ordering optimization techniques. Since the entities those are not linked with any other entity would still be a part of final dataset, full outer join optimization algorithms would be necessary to decide on the order.

One subtle issue here is to identify selectivity (i.e. the ratio of intersecting entities to all entities of the given two datasets) of dataset pairs. Here, an estimate of the selectivity would be enough to rank dataset pairs. Intuitively, two mobility datasets that have higher number of common entities should have higher spatio-temporal overlap of records. Since the idea behind the LSH is exploiting the same property of the datasets, applying LSH to all datasets and using the ratio of number of candidate pairs to number of all pairs of entities as the estimated intersection ratio of given two datasets would give be an indicator of high intersection. With link and merge functions satisfying ICAR properties and

estimated selectivity between datasets, any join ordering optimization algorithm (i.e. Greedy Operator Ordering [45]) would create a join ordering that completes the process in optimum running time.

However, in the existence of negative evidences (i.e. alibi) representativity cannot be satisfied. Moreover, since in every pairwise linkage would create a one to one linkage of entities, associativity cannot be guaranteed as well. Therefore, leveraging aforementioned process with described link and merge functions is scalable but cannot guarantee the completeness of the result. We believe this is an important problem and leave it as future work.

Chapter 6

Experimental Evaluation

In this chapter we present our experimental evaluation. In the first section we present an evaluation of the proposed k-l diversity based linkage method and the *ST-Link* algorithm. It contains two sets of experiments. In the first set of experiments in the section, we measure the performance and the scalability of the *ST-Link* algorithm. By increasing the size of the input data, we test the change in the running time, the number of event comparisons, and the number of candidate user pairs, for different window sizes. In the second set of experiments, we analyze the quality of the k-l diversity based linkage. To measure quality, we use two metrics. The first is the precision, which measures the fraction of correctly linked pairs in the list of user pairs produced by *ST-Link*. The second is the number of true positives, which is the number of user pairs correctly linked by the *ST-Link* algorithm.

In the second section, we present an evaluation of similarity based approach and *SLIM* algorithm. This evaluation also includes two sets of experiments. In the first set of experiments, we measure the quality of the linkage and the *SLIM* algorithm's robustness under changing spatio-temporal levels, record densities, and the entity intersection ratio across the two datasets. To test the quality, we use two measures: precision and recall. Moreover, to better understand the behaviour of *SLIM* when the aforementioned variables are changing, we also

present the alibi counts and the number of pairwise record comparisons. In the second set, we measure the scalability achieved via LSH and its effect on the quality. We measure the number of pairwise entity comparisons and the $F1$-Score of the resulting linkage, by varying the parameters of the LSH procedure for mobility histories, i.e., the similarity threshold t, the query time window size, and the spatial level of the dominating grid cells.

In the third section, we compare *ST-Link* and *SLIM* with the state-of-art. First, we compare *ST-Link* with SERF – Stanford Entity Resolution Framework that implements the R-Swoosh [24] algorithm. Next, we compare *ST-Link* and *SLIM* with two existing approaches, namely, Pois [14], and GM [19] in terms of the $F1$-Score, Hit Precision @40, and the number of pairwise record comparisons. We incorporate a new measure, Hit Precision @40, because it is the original measure used by the authors of GM.

6.1 Evaluation of Rule Based Linkage

In this section, we present an evaluation of the proposed k-l diversity based linkage method and the *ST-Link* algorithm. We implemented the *ST-Link* algorithm using Java 1.7. All experiments were executed on a Linux server with 2 Intel Xeon E5520 2.27GHz CPUs and 64GB of RAM.

We present two sets of experiments. In the first set of experiments, we measure the performance and the scalability of the *ST-Link* algorithm. By increasing the size of the input data, we test the change in the running time, number of event comparisons, and the number of candidate user pairs, for different window sizes. In the second set of experiments, we analyze the quality of the k-l diversity based linkage. To measure quality, we use two metrics. The first is the precision, which measures the fraction of correctly linked pairs in the list of user pairs produced by *ST-Link*. The second is the number of true positives, which is the number of user pairs correctly linked by the *ST-Link* algorithm.

6.1.1 Datasets Used

For the performance, scalability, and accuracy evaluations we used three datasets. The first is a Foursquare dataset of check-ins. The second is anonymized call detail records in a telecommunication provider. For privacy concerns, we did not perform any linkage across these two datasets. As a result, we were not able to compute accuracy results when using these two datasets. However, they are used for the evaluation of running time performance. To evaluate accuracy, linkage is performed between a third dataset belonging to a hypothetical LES and the call dataset. This dataset was synthetically derived to protect privacy, from the call dataset by (i) picking a predefined fraction f of the callers at random as active users of the second LES, (ii) generating usage records for the selected users by assuming that they generate such a record with probability p, within a 15 minute time window of a call, inside a location within the same cell tower of the call. We change the parameters p and f to experiment with different scenarios. Lower values for p result in a sparser usage record dataset for the second LES. We call the parameter p, the *check-in probability*. As not all users have the same check-in probability in practice, we pick the value of the check-in probability for a given user from a Gaussian distribution with mean p. We call the parameter f, the *usage ratio*.

Datasets ⇒	Foursquare	Call
# of activities	1,903,674	1,890,107,057
# of venues/cell towers	300,685	109,780
# of users	284,856	3,357,069

Table 6.1: Dataset statistics (ST-Link experiments)

The Foursquare dataset consists of check-ins that were shared publicly on Twitter, collected via the Twitter streaming API[1] and the Foursquare API[2]. This dataset spans 40 days and only contains check-ins from Country X. Each row contains the acting user's Foursquare id, venue id, geographical location (lat/lon) of the venue, and the time of the check-in. The call dataset spans the

[1] www.dev.twitter.com
[2] www.developer.foursquare.com

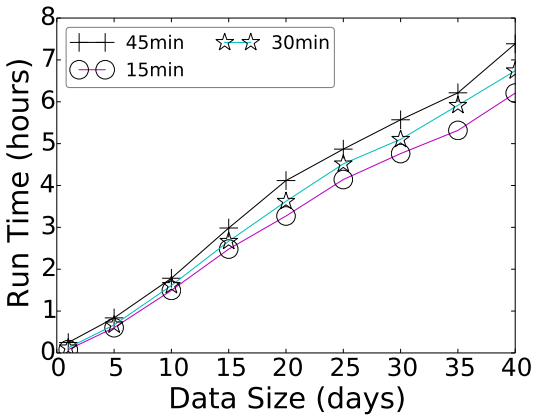


Figure 6.1: Running time vs. dataset size.

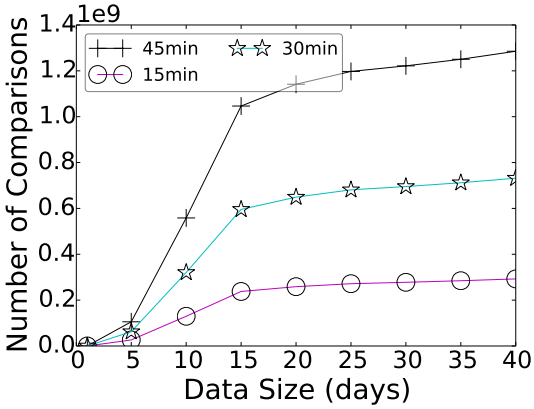


Figure 6.2: Number of comparisons vs. dataset size.

Figure 6.3: Performance Results

same 40 days in Country X. Each row contains an anonymized id, time of the call, and geographical location (lat/lon) of the handling cell tower. The anonymized id is the same across all usage of the same user. Table 6.1 shows the statistics about both the Foursquare and the call datasets. For the runtime performance and filtering effectiveness experiments, we used the two real datasets. However, since it is not possible to verify the accuracy of the results using these two datasets, we used the synthetic dataset which is derived from the anonymous call data for the evaluation of *ST-Link*'s accuracy.

6.1.2 Running Time Performance

We observe the running time, the number of candidate pairs, and the number of event comparisons as a function of the dataset size. The dataset size is increased by increasing the number of days of data included in the linkage analysis. Furthermore, for these experiments we also change the window size. Recall that the window size is used during the temporal filtering step to locate co-occurring events.

Figure 6.6 presents our running time related results. In all the figures, the x-axis represents the dataset size in days and the y-axis represents a performance

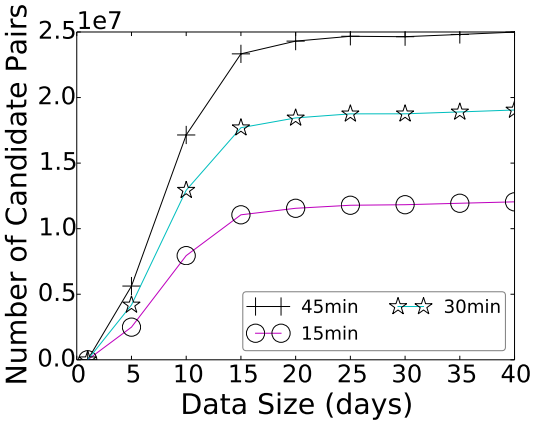


Figure 6.4: Number of candidate user pairs vs. dataset size.

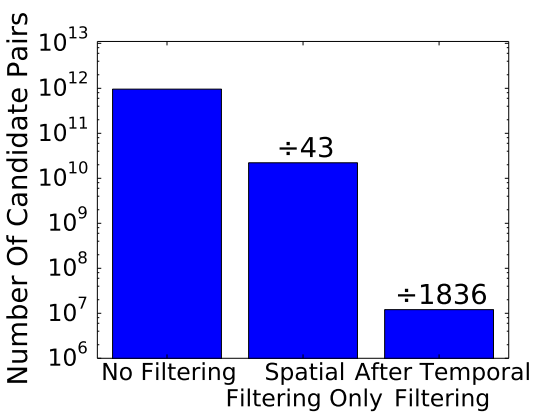


Figure 6.5: Reduction in the number of possible pairs.

Figure 6.6: Performance Results

metric. Different series represent varying window sizes.

One of the main challenges is the scalability of the linkage solution. Processing many days of data should complete in reasonable amount of time for the resulting analysis to be valuable. Figure 6.1 plots the running time as a function of the dataset size. We make two observations from the figure. First, the running time of *ST-Link* is linear in the dataset size. For 5 days of data, the running time is around 1 hour and for 40 days of data it is around 7 hours, all for 30 minute windows. Second, the running time increases with increasing window size, yet the running time is linear in the dataset size for all window sizes.

Figure 6.2 plots the number of event-to-event comparisons as a function of the dataset size. In our experimental evaluation, every time we compare two location based events for either co-location or alibi check, the number of event comparisons is increased by one. We observe that up to 15 days of data, the number of comparisons grows at an increasing rate. Yet, after 15 days the rate starts to go down and eventually the growth of the number of operations happens at a relatively low fixed rate. This can be explained by the alibi checks performed by *ST-Link*. Recall that when a user pair is marked as an alibi, their records are not compared with each other anymore. Also, if two users are marked as a candidate pair, their future records are only compared to see if they are an alibi

or not. Considering this, we can say that within 15 days most of the candidate pairs and alibi pairs are identified. As an important difference from the running time experiment, the gaps between the series corresponding to the three window sizes are considerably larger. This is because larger windows require more event to event comparisons. Since event comparisons are not necessarily the only cost of the algorithm (there is I/O, window processing, window index maintenance, etc.), the running time experiment has narrower gaps between the running times for different window sizes. The impact of these extra costs can be seen in Figure 6.1 as well; although the number of comparisons stabilize after 15 days, the linear increase in the runtime continues.

Figure 6.4 plots the number of candidate user pairs as a function of the dataset size. Just like for the number of comparisons experiment, up to 15 days, the number of candidate pairs grows with an increasing rate and after 15 days the rate starts to decrease and eventually stabilizes at a low value. For the case of candidate pairs, the eventual rate of increase is very low, suggesting that observing additional data brings diminishing returns in terms of being able to find new candidate pairs. However, this does not imply that we are unable to perform additional linkages, because the number of linked pairs within the candidate set can still grow (we will observe such growth in the quality experiments).

Figure 6.5 shows the number of candidate user pairs after each filtering step. It illustrates the effectiveness of the spatial and temporal filtering steps of *ST-Link*. If no filtering was applied on the data, every user pair from the two datasets would have constituted a candidate user pair. By applying only spatial filtering, the number of candidate user pairs decreases by 43 times compared to the no filtering case. It is possible to say that spatial filtering is an effective step. Intuitively, if data was spread over a wider geographical area, this step would be have been even more effective (our datasets are limited to the geographic area of Country X). After applying temporal filtering, the candidate user set decreases by an additional $1,836$ times after spatial filtering. Cumulatively, the number of candidate user pairs without any filtering is $78,948$ times of the number of pruned candidate user pairs, which gets close to 5 orders of magnitude reduction in the number of pairs.

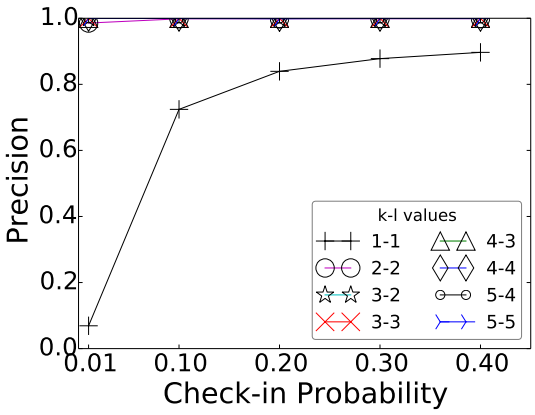


Figure 6.7: Precision as a function of check-in probability.

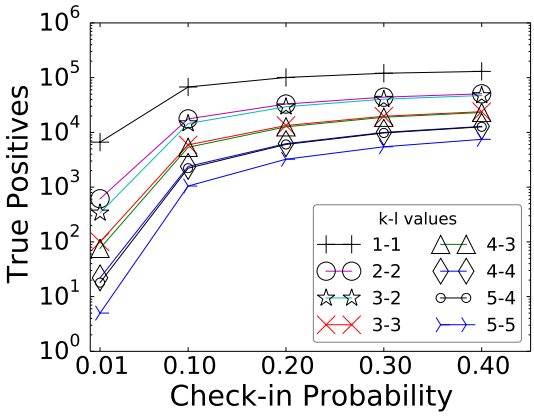


Figure 6.8: Number of true positives as a function of check-in probability.

6.1.3 Quality of Linkage

We observe the precision and the number of true positives as a function of the usage ratio, check-in probability, and window size. We also observe the precision and recall values for a variation of the *ST-Link* algorithm that does not use weights, thus trades off precision for better recall.

6.1.3.1 Impact of Check-in Probability

Figures 6.7 and 6.8 plot the precision and number of true positives, respectively, for the results produced by the *ST-Link* algorithm as a function of the mean check-in probability. Different series in the figure represent different k-l settings. We set $k \geq l$, as the co-occurrence counts has to be greater than the diversity counts. For these experiments the usage ratio is set to 50%, which means only half of the call users are performing check-ins.

Figure 6.7 shows that precision is very close to 1 for all k-l settings but for 1-1. We see that using 1-1 diversity results in very poor precision for low values of the check-in probability. As the check-in probability increases, then the precision of 1-1 diversity increases as well, but never reaches 1. The increase is understandable, as more events on the check-in side will help rule out incorrect candidate pairs

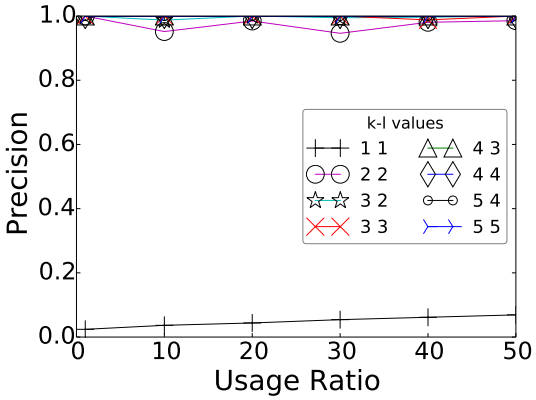


Figure 6.9: Precision as a function of usage ratio.

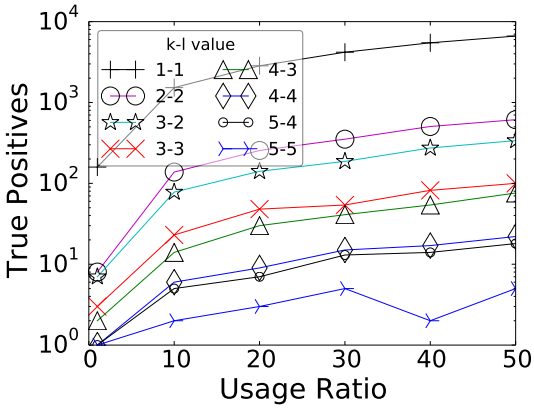


Figure 6.10: Number of true positives as a function of usage ratio.

via alibis. Surprisingly, the precision for higher k-l values are all close to 1. This is due the impact of alibi detection, and strong weight constraint. As we will see shortly, not using weights trades off precision for better recall. Even if two users have events that are only 2-2 diverse, they can be correctly linked if they have no alibis.

Figure 6.8 shows that the number of true positives in the linkage increases with the increasing check-in probability. This is expected, as more events help in increasing the co-occurrence and diversity counts. We also observe that higher k-l values result in reduced number of linkages. Given that 2-2 diversity has very good precision, and has the second highest true positive count (after 1-1 diversity, which has unacceptable precision), it can be considered a good setting for getting the best out of the linkage. We see that for a check-in probability as low as 0.01, it can match many hundreds of users, and for probability 0.1, it can match up to 10 thousand users. As we will see shortly, these numbers can be further increased by trading off some accuracy.

6.1.3.2 Impact of Usage Ratio

Figures 6.9 and 6.10 plot the precision and the number of true positives, respectively, for the results produced by the *ST-Link* algorithm as a function of

	Precision	Recall
2-2	0.19	0.31
3-2	0.36	0.32
3-3	0.89	0.61
4-3	0.93	0.47
4-4	0.99	0.58
5-4	0.99	0.47
5-5	0.99	0.50

Figure 6.11: Precision and recall using unweighted linkage

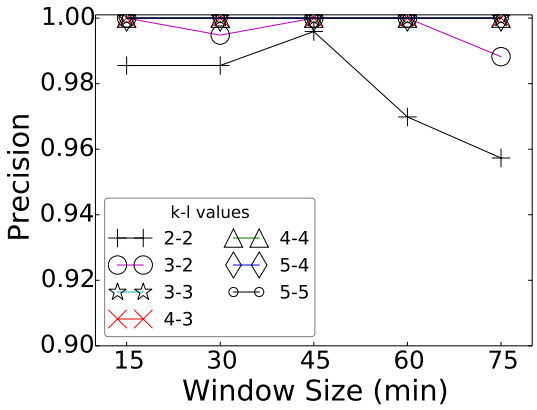


Figure 6.12: Precision as a function of window size.

the usage ratio. Different series in the figure represent different k-l settings, as before. For these experiments check-in probability is taken as 0.01.

Figure 6.10 shows that the precision of all k-l settings is close to 1 throughout the entire range of the usage ratio, except for 1-1. The 2-2 setting has precision values that are slightly lower than 1, but not lower than 0.95. Figure 6.8 shows the true positive counts for the same settings. As we can see clearly from the figure, increased usage ratio results in increased number of successful linkages. Again, this could be attributed to increasing weights for co-occurrence and diversity, as well as increased effectiveness of alibi detection.

Interestingly, even when only 1 percent of the call users are synthetically set to making check-ins, and when the check-in probability around a call is set as low as 1 in 100, one can still match some users (around 10). This could also be looked at from a privacy standpoint. In other words, being able to perform spatio-temporal linkage across two datasets successfully even for only 10 users may be considered a privacy breach. We plan to investigate the privacy protection mechanisms against this kind of linkage in our future work.

6.1.3.3 Unweighted Linkage

The results so far have demonstrated high precision, but the number of users one could match is relatively low compared to the number of total users. In order to show the trade-off between precision and accuracy, we have also performed experiments where the linkage model is slightly modified to use weights that are equal to 1. That is, we count each event co-occurrence between two users as 1, without considering other possible co-occurrences these events may have with events of other users. In other words, the weight function from Eq. 3.9 is taken as 1. As it was discussed before, there are two different approaches for deciding the values of the k-l parameters. The first one is deciding after multiple experimental runs, and the second one is by detecting the *elbow point* of distributions of co-occurrence and diversity values.

In this experiment we applied both to show the effectiveness of the *elbow point* detection technique as well. According to maximum absolute second derivative test results, the values of k-l parameters based on elbow detection are 3-3.

Table 6.11 shows the precision and recall results for the unweighted linkage. The recall values here represent the fraction of users from the check-in dataset that were successfully linked. It is important to note that we only considered users that have enough number of events. Users are said to have enough number of events only when they have at least l diverse events, for each k-l setting. The table shows an interesting result: With unweighted linkage we see a clear trade-off, where with increasing k-l values the precision improves, but the recall drops. With the 3-3 setting, we get a precision of 0.89 and can link 61% of the users that have enough number of events. Considering all users from the check-in dataset this value is 23 %. Recall that 3-3 setting was identified using the *elbow point*s. Increasing the diversity setting to 5-5, one gets almost perfect accuracy (0.99), but the recall drops to 50% of the users. When absolute accuracy is not required, such as for machine learning to extract overall patterns, the unweighted linkage model could be more effective in practice.

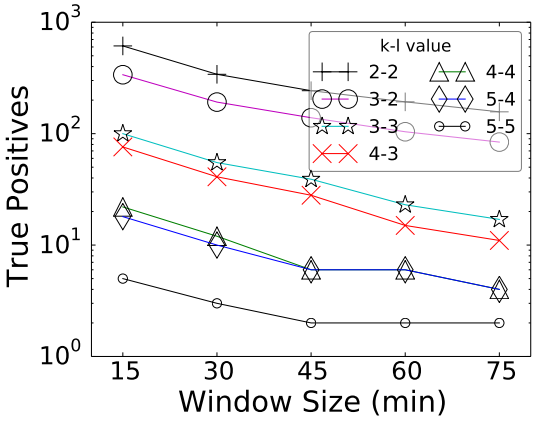


Figure 6.13: Number of true positives as a function of window size.

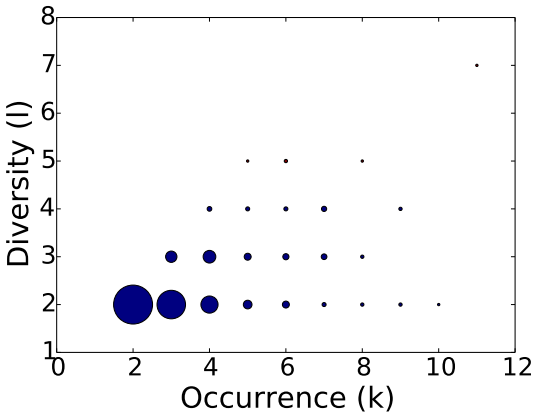


Figure 6.14: k-l values distribution

	Runtime (m)	Precision	Recall	Cand. Count
1	30	0.78	0.68	1,998,491
2	34	0.75	0.74	2,651,746
4	39	0.71	0.82	3,511,090
8	45	0.68	0.88	4,446,937
16	58	0.65	0.91	5,311,043
∞	122	0.62	0.99	6,765,345

Figure 6.15: Alibi threshold experiment results.

6.1.3.4 Impact of Alibi

Alibis are used to improve both the running time performance and the accuracy. For these experiments the check-in probability is taken as 0.5 and the usage ratio is taken as 50%. Only one grid is considered, which contains 15,268 users and 1,956,734 events in total. As it was discussed before, a threshold value on the number of alibi events can be used before disregarding a candidate pair. In this experiment we evaluate the impact of alibi in terms of performance and accuracy as a function of the alibi threshold.

Performance. Table 6.15 shows the running time, precision, recall, and the number of candidate pairs for the alibi threshold experiment. When the threshold is set to ∞, effectively disabling alibi detection, we observed that the algorithm

took 122 minutes to complete. At the end of the temporal filtering step, there were 6,765,345 possible pairs. On the other hand, when alibi is used and the threshold is set to 1, the running time decreased down to 30 minutes and the number of possible pairs were 1,998,491. Almost 70 % of the possible pairs were pruned with the help of alibi detection and further processing is avoided. When larger threshold values are used, we observe slight increase in the running time. For the threshold values of 2 and 16, the processing time is 34 and 58 minutes, respectively.

Accuracy. Precision of the k-l diversity based linkage can be increased by setting sufficiently large k and l values. Larger k and l values decrease the probability of different users satisfying the linkage requirements. However when at least one of the datasets is sparse, setting larger k and l values will result in low recall, as many true positive pairs will be missed. In such datasets, alibi definition prevents many false positive pairs that satisfy the co-occurrence and diversity requirements. Our experiments showed that when alibi is not used (threshold value ∞), 99% recall can be reached, yet with 62% precision. In contrast, setting alibi threshold to 1, increases the precision to 78%, with recall decreased to 68%. The reason behind this decrease has to do with the lack of precise location information in our datasets. For example, when two temporally close events of a user are from two neighboring cell towers, their locations end up being the centers of the cell towers, as the location information is not sufficiently fine grained. This results in incorrectly identifying a pair of events as alibis, as the distance between the event locations is relatively high when considering their close timestamps. This is when the alibi threshold becomes crucial. We observe that the recall increases to 0.74% when alibi threshold set to 2. Increasing threshold further increases the recall values with a cost of sacrificed precision. For alibi threshold 4 recall is 0.82% and precision is 71%.

6.1.3.5 Window Size

Figures 6.12 and 6.13 plot the precision and the number of true positives, respectively, as a function of the window size. Different series in the figure represent

different k-l settings, as before. For these experiments the check-in probability is taken as 0.01 and the usage ratio is taken as 50%. Window sizes start from 15 minutes and increases up to 75 minutes in increments of 15 minutes.

Figure 6.12 shows that the precision stays at 1 is not effected by the window size except for lines corresponding to lower k-l values. In particular 2-2 and 3-2 are impacted negatively from larger window sizes. 1-1 is not shown in this experiment, as it already has a very low precision. Note that the window size does not impact only the size the temporal window we slide over the events, but also the definition of co-occurrence (recall the α parameter from Eq. 3.2). Increasing the window size makes it possible to match potentially unrelated events from different real-world users and the results reflect that. However, due to the alibi processing, the negative impact of increasing window size on the precision is milder that it would otherwise be.

Figure 6.13 shows that the number of true positives drops with increasing window sizes. Again this can be attributed to the increasing number of unrelated event matches due to the larger window. Recall that if the same user is matched to more than one user from the other dataset, we remove such users from the linkage results. The increased window size results in ambiguity in the results. Assuming users x and y are linked for a given window size, increasing the window size does not change the fact that x and y are matched, but it may result in additional matches, such as between user x and some other user z, and thus eliminating the correct linkage between x and y from the results.

6.1.3.6 k-l value Distribution

Figure 6.14 shows the k-l value distribution of user pairs after spatial and temporal linkage. The usage ratio is taken as 50% and the check-in probability is 0.01 for this experiment. For a given pair in the results, we find the highest k-l diversity values it supports and maintain these counts. In the figure, the areas of the circles are proportional to number of pairs with the given k-l diversity. Since the number of pairs for k-l lower than 2-2 is too high (and precision very low as

we have seen earlier), we do not present them in the results. As expected, the number of linked pairs is decreasing as the k-l values are increasing. It is interesting to note that for extreme values such as 11-7 diversity, it is still possible to find user pairs. We also observe that increasing diversity has a higher filtering power than increasing occurrence.

6.2 Evaluation of Similarity Based Linkage

In this section, we give a detailed evaluation of the proposed solution, *SLIM* and compare it with state-of-the-art. We implemented *SLIM* using Java 1.8. All experiments were executed on a Linux server with 2 Intel Xeon E5520 2.27GHz CPUs and 64GB of RAM. Evaluation includes two sets of experiments. In the first set of experiments, we measure the quality of the linkage and the *SLIM* algorithm's robustness under changing spatio-temporal levels, record densities, and the entity intersection ratio across the two datasets. To test the quality, we use two measures: precision and recall. Moreover, to better understand the behaviour of *SLIM* when the aforementioned variables are changing, we also present alibi counts and the number of pairwise record comparisons.

In the second set, we measure the scalability achieved via LSH and its effect on the quality. We measure the number of pairwise entity comparisons and the $F1$-Score of the resulting linkage, by varying the parameters of the LSH procedure for mobility histories, i.e., the similarity threshold t, the query time window size, and the spatial level of the dominating grid cells.

6.2.1 Datasets

We use real-world datasets for two major linkage experiments. The first set of linkage experiments is performed using the *Cab* dataset which contains mobility traces of approximately 530 taxis collected over 24 days (from 2008-05-17 to 2008-06-10) in the San Francisco Bay Area. It has $11,073,781$ records in total.

The second set of experiments is performed on linking *Social Media* data, which contain records from Foursquare and Twitter shared publicly on Twitter. This *Social Media* (*SM*) dataset contains around 5 million records: $2,266,608$ records from $197,474$ Twitter users and $2,987,747$ records from $276,796$ Foursquare users. The dataset spans 26 days from 2017-10-03 to 2017-10-29 and location information is distributed over the globe. We give these details also in Table 6.2

Datasets $\Rightarrow$	Foursquare	Twitter	Taxi
# of entities	276,796	197,474	530
# of events	2,987,747	2,266,608	11,073,781
# of days	26	26	24

Table 6.2: Dataset statistics (SLIM Experiments)

For maintaining privacy, we generated a unique identifier for each user, and removed all personally identifying information. [3] As the *SM* dataset has one-to-one correspondence in many records (i.e., a check-in is also shared in Twitter as a tweet), we increase the complexity of the linkage by using Twitter and Foursquare as base sources and sampling them.

To control the experimental setup, we use two parameters during this sampling: *entity intersection ratio* and *record inclusion probability*. The intersection ratio is used to control the ratio of the entities common in both datasets. This is incorporated in subset generation as follows. We first randomly split all entities into two disjoint datasets of equal number of entities. One of these sets is used as the *pivot* dataset. For a given intersection ratio, we randomly select the corresponding number of entities from the pivot dataset and replace the same number of randomly selected entities from the second dataset with these. Once the entities are finalized, we also downsample the records from the datasets. This is to address the common case in real life that two location-based services are not always used synchronously in practice and different services might have different usage frequencies. A record of an entity is included in a dataset with the record inclusion probability. Higher probability implies denser datasets. With this approach, we have datasets with different record densities and with different

[3] The data collection and use for this study received approval from Bilkent University's ethics board.

amounts of entity overlap. After this step, we link samples from Twitter dataset with samples from Foursquare dataset (referred as SM together). The same applies to Cab dataset. We sample the data with different entity intersection ratio and record inclusion probability values and link those sampled datasets to each other.

The default values for the entity intersection ratio and the record inclusion probability are both equal to 0.5. The spatial detail at the mobility history leaves are controlled using the cell levels of S2. A higher level indicates more spatial detail.The default value for the spatial level is 12, and the default temporal window width is 15 minutes. To avoid the adverse effect of entities with too small number of records after downsampling, we ignore an entity if it does not have more than 5 records. The default value of the parameter b from Equation 4.2 is 0.5. To identify the alibi threshold, we set the maximum movement speed of an entity to 2 km/minute and multiply this constant with the temporal window width.

6.2.2 Accuracy

In this section, we first study precision and recall as a composite function of the spatio-temporal level. We also look at the number of alibi entity pairs and the number of pairwise record comparisons to better understand SLIM's behavior. Next, we study $F1$-Score and running time as a function of the record inclusion probability.

6.2.2.1 Effect of the Spatio-Temporal Level

Figures 6.20 and 6.25 plot precision, recall, alibi pairs and number of record comparisons as a function of the spatio-temporal level for the Cab and SM datasets, respectively. In all figures, the x-axis shows the spatial detail, the y-axis shows the width of the window in minutes, and the z-axis shows the measure.

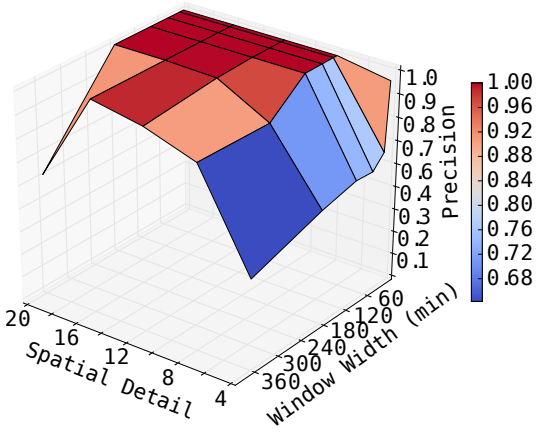


Figure 6.16: Precision

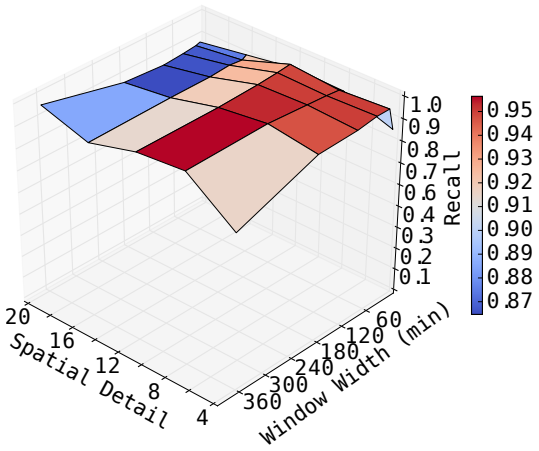


Figure 6.17: Recall

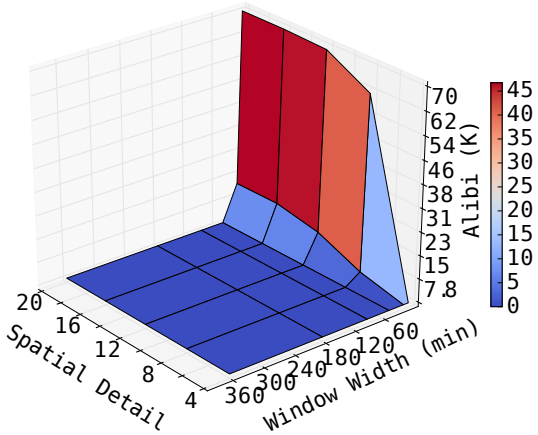


Figure 6.18: # of alibi pairs

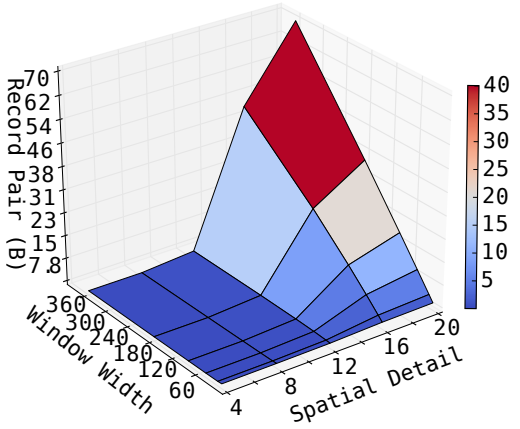


Figure 6.19: # of event comparisons

Figure 6.20: Effect of the spatio-temporal level – Cab

Figures 6.16 and 6.17 show precision and recall for the Cab dataset. We observe that both measures increase with spatial detail. This is because when the spatial detail increases, the distance calculation becomes more accurate.

After spatial level 12, $F1$-Score becomes greater than 0.95. However, for window width, after 90 minutes, while recall remains high, precision decreases dramatically. For spatial detail 20, when the window size is 15 minutes, perfect precision is reached, while for window size 360 minutes the precision is 0.56. The decrease in precision is steeper for spatial detail 20 than spatial detail 16, but for

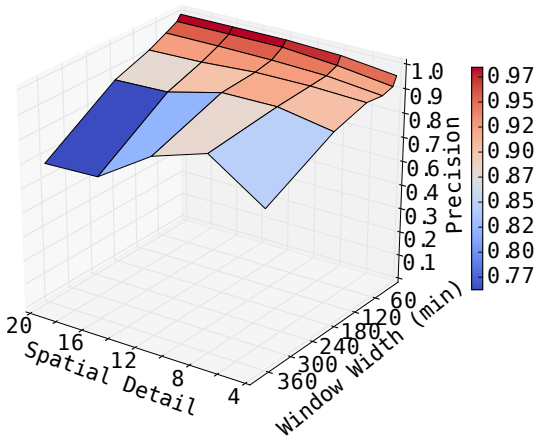


Figure 6.21: Precision

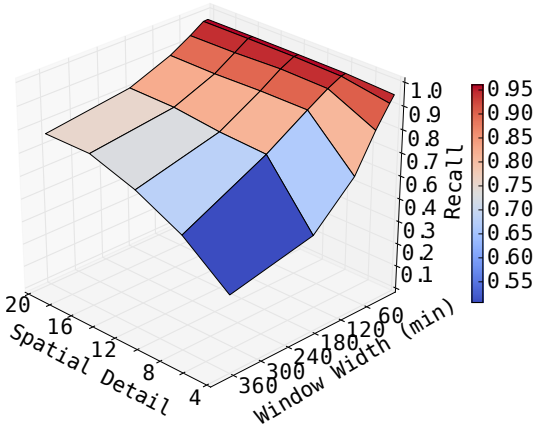


Figure 6.22: Recall

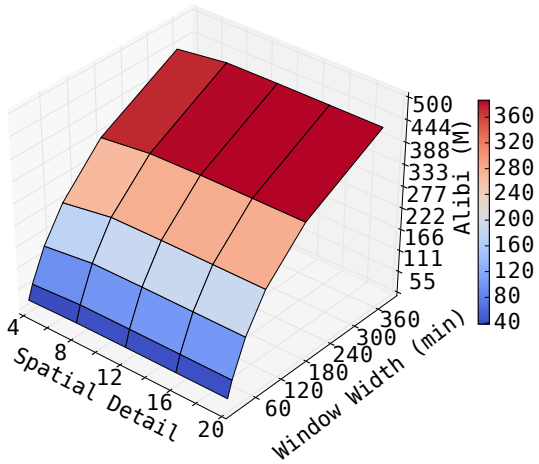


Figure 6.23: # of alibi pairs

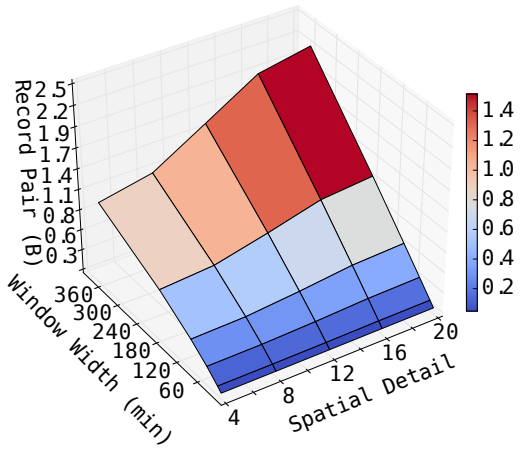


Figure 6.24: # of event comparisons

Figure 6.25: Effect of the spatio-temporal level – SM

the same data point recall is higher for spatial detail 20 than 16. The reason behind this is that, since the records in the same time-location bins are aggregated, using large time-windows makes it harder to distinguish entities from each other. When the level of detail is low, both spatially and temporally, variance of entity pair scores is decreasing. To observe this behaviour better, figure 6.26 shows detected stop threshold values (red lines), fit GMM models (blue and green curves), and distribution of true positive and false positive links for spatial detail values 4, 8, 12, and 16 as a function of similarity scores for window width 90 minutes. We observe that with increasing spatial detail, grouping true positive links (green bars) and false positive links (blue bars) in two clusters becomes more accurate

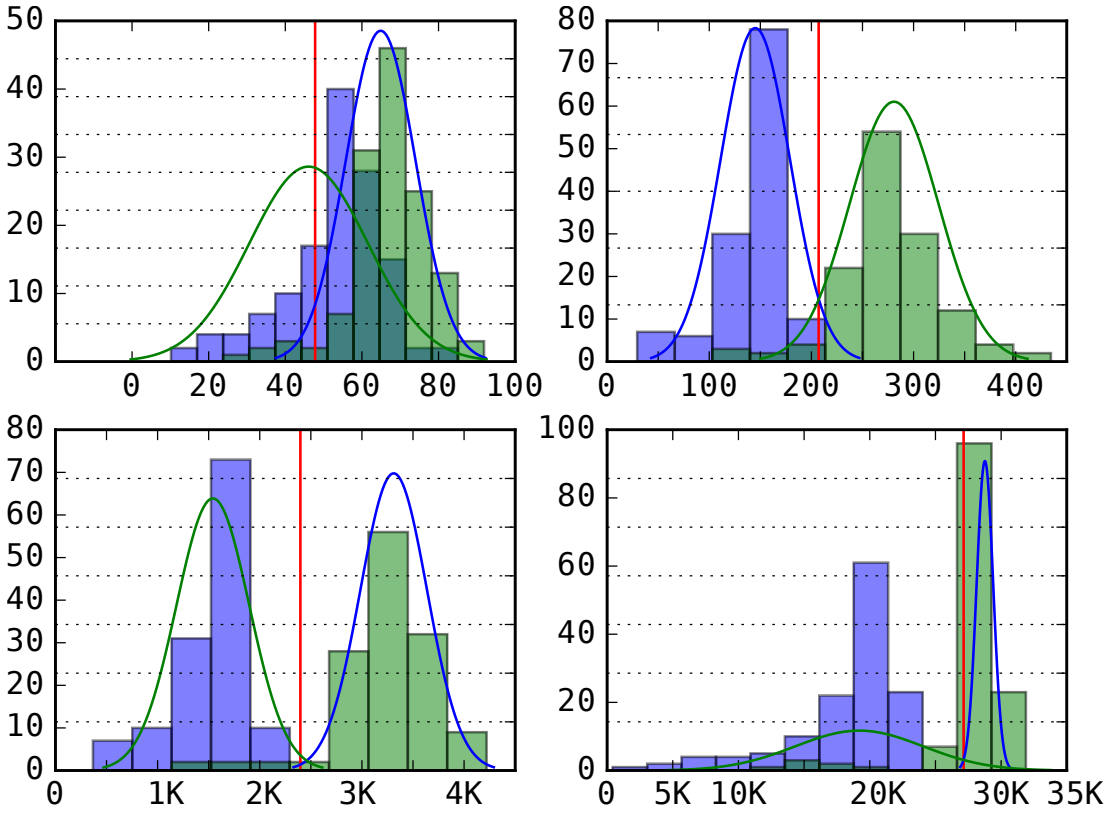


Figure 6.26: Similarity score histograms

and a tighter stop threshold value could be identified. By looking at the distances between two components of GMM one could say that stop threshold identification has subpar accuracy for spatial detail values lower than 12. One could observe this subpar accuracy by looking at the differences between precision behaviours of Figures 6.16 and 6.21. For low spatial detail (i.e., ≤ 10) and high temporal window width (i.e., ≥ 60 minutes), precision is favored over recall for the Cab dataset but vice-versa for SM.

While spatial level values higher than 12 have similar precision and recall, we observe that increasing spatial detail also increases the number of record comparisons. This is expected, as we discussed in Section 4.3.4 how to use the trade-off between accuracy and performance to detect the best spatial detail for a given temporal window. When the window size is 15 minutes, the spatial detail detected by the parameter tuning algorithm is 12. Figure 6.19 shows that for the same window width, increasing spatial detail from 12 to 20 increases the number

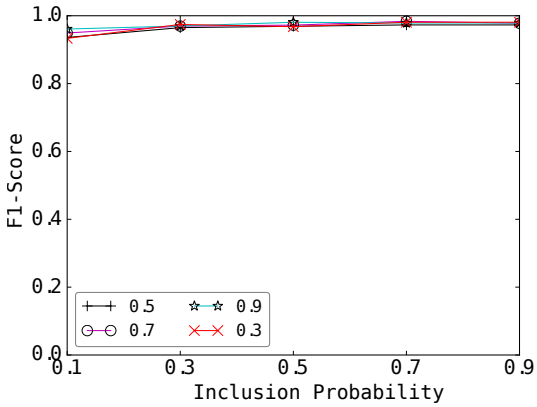


Figure 6.27: F1-Score – Cab

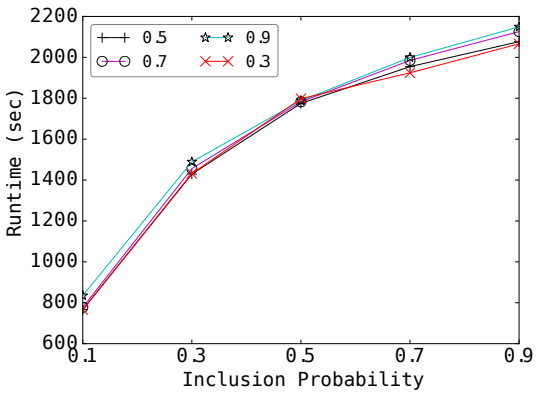


Figure 6.28: Runtime – Cab

of pairwise record comparisons by 1.14 times, yet the accuracy stays the same. The gap widens for longer temporal windows. The same figure shows 3.15× increase in the number of record comparisons when the window size is increased from 15 to 360 minutes, for spatial detail 12. Yet, the increase is 22× for spatial detail 20.

Figure 6.25 shows the same experiment for the SM dataset. Most of the previous observations hold for this dataset as well. There are two additional observations. First, in the Cab dataset the best recall value is reached when 5-minute windows are used. This is a result of the spatio-temporal density of the Cab dataset, as alibi detection becomes more efficient for narrow windows, resulting in better recall. On the other hand, in the SM dataset, the best recall is reached for 15-minute windows. This is expected, because at one extreme very small temporal windows require services to be used synchronously to collect evidence for linkage. Another observation is that, as SM dataset has lower spatio-temporal skew, to detect alibis one needs to use larger temporal windows.

6.2.2.2 Sensitivity to the Workload Parameters

In this experiment, we link the pivot datasets (with record inclusion probability of 0.5) of each source with the datasets generated using different entity intersection ratios and record inclusion probabilities. Figure 6.31 plots the $F1$-Score and

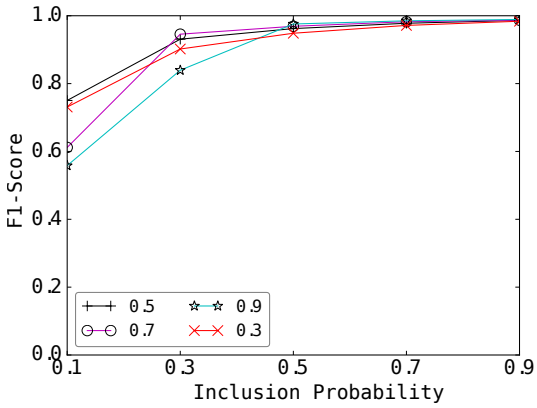


Figure 6.29: F1-Score – SM

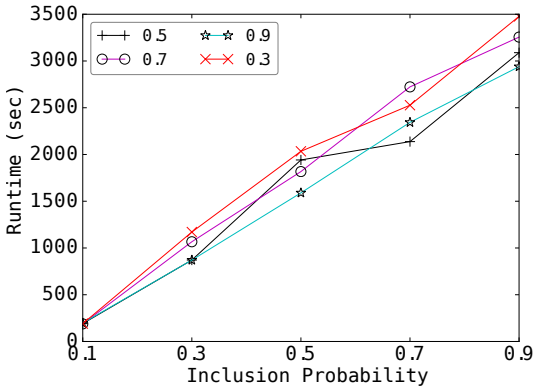


Figure 6.30: Runtime – SM

Figure 6.31: F1-Score and Runtime as a function of the inclusion probability (for different entity intersection ratios)

running time in seconds as a function of record inclusion probability for the Cab and SM datasets, respectively. Different series represent different entity intersection ratios. The Cab dataset has 265 entities and the SM dataset has around $30,000$ entities. The average number of records for an entity is ranging from $2,100$ to $18,900$ for the Cab dataset and from 10 to 20 for the SM dataset. With this structure, this experiment also helps us to illustrate scalability of the linkage with increasing number of records. Figure 6.27 shows the results for the Cab dataset. We observe that all $F1$-Score values are close to 1, even when average number of records are as low as $2,100$ (inclusion probability 0.1). Moreover, from Figure 6.28, we observe that the running time is sub-linear with average number of records, which is a result of aggregation on mobility histories. These results validate robustness of SLIM, as $F1$-Score is not effected by the increasing number of records and the system scales linearly.

Figures 6.29 and 6.30 show the results of the same experiment for the SM dataset. Different than the Cab dataset, the effect of the record inclusion probability on the $F1$-Score is more pronounced here. For the entity intersection ratio value of 0.5, the average number of records is 10, $F1$-Score is 0.75. When the average number of records is doubled, we get 0.98 as the $F1$-Score. This is because the average number of records per entity is already low in the SM dataset and downsampling it to even lower values decreases the number of records that can

serve as evidence for linkage. However, we observe that *SLIM* is able to perform high-accuracy linkage when the average number of records per entity is at least 15. Independent of the entity intersection ratio, after 15 records per entity, the $F1$-Score of *SLIM* is greater than 0.9. Similar to Cab dataset, running time of *SLIM* is linear with the input size for SM dataset.

6.2.3 Scalability

In this set of experiments, we study the effect of the LSH on the quality and the scalability of the linkage. The quality is measured using the $F1$-Score relative to that of the brute force linkage. Let $F1$-Score where LSH is applied be $F1\text{-Score}_{lsh}$ and without it $F1\text{-Score}_{bf}$. Then, relative $F1$-Score equals $F1\text{-Score}_{lsh}$ / $F1\text{-Score}_{bf}$. To measure the speed-up, we compute the ratio of the number of pairwise record comparisons without LSH to that of with LSH. Let the number of pairwise record comparisons with LSH be C_{lsh} and without it be C_{bf}. Then, the relative speed up equals C_{bf} / C_{lsh}. We split this set of experiments into two subsets. In the first subset, we study the effect of the spatio-temporal level of the dominating cell calculation algorithm. In the second subset, we study the effect of the LSH parameters, namely similarity threshold t and the number of buckets.

6.2.3.1 Effect of the Spatio-Temporal Level

Figure 6.36 shows the relative $F1$-Score and the speed-up as a composite function of the spatial level and the temporal step size. Recall that we construct a set of dominating cells to act as signatures. This construction is done by querying each mobility history for non-overlapping time windows. The size of each grid cell is defined by the spatial level. The temporal step size represents the number of time windows the query spans. Note that these parameters are different than the spatial level and the window width that is used for the similarity score computation. The LSH similarity threshold t is set to 0.6 and the number of buckets is set to 4096.

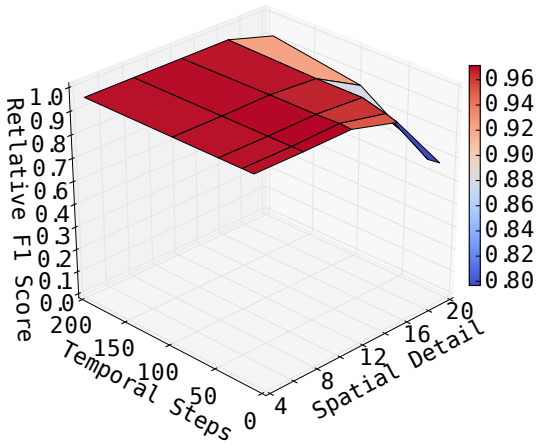


Figure 6.32: $F1$-Score – Cab

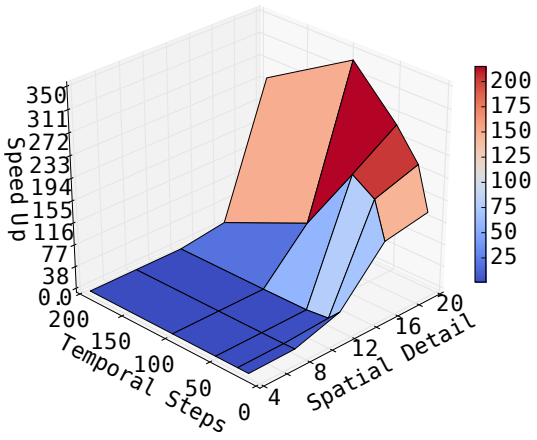


Figure 6.33: Speed-up – Cab

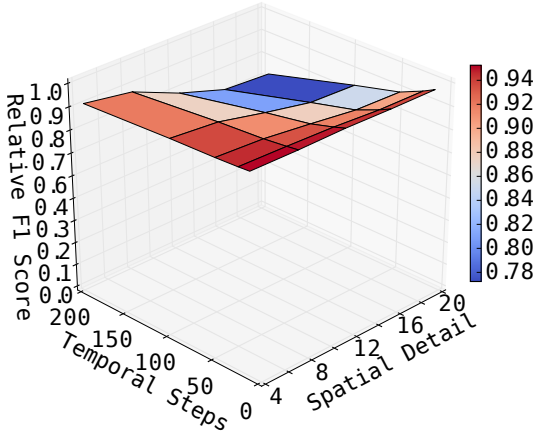


Figure 6.34: $F1$-Score – SM

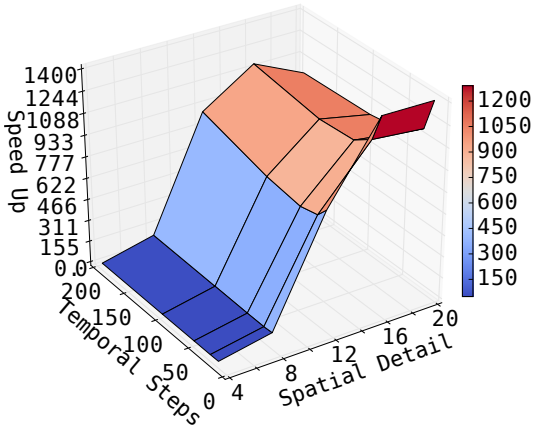


Figure 6.35: Speed-up – SM

Figure 6.36: LSH accuracy and speed-up as a function of the spatial level and temporal step size

Figures 6.32 and 6.33 show the relative $F1$-Score and speed-up, respectively, for the Cab dataset. Figure 6.32 shows that the $F1$-Score achieved with and without LSH are almost the same when the spatial level is lower than 12. Similarly, we do not observe any speed-up for these data points. The reason is that the Cab dataset is spatially too dense and consequently dominating grid cells of all entities end up being the same when the spatial detail is low. However, when the spatial detail is increased, we observe that LSH brings 2 orders of magnitude speed-up by preserving 98% of the $F1$-Score. For the spatial detail value of 16 and temporal step size of 48 (this means each dominating grid cell query spans

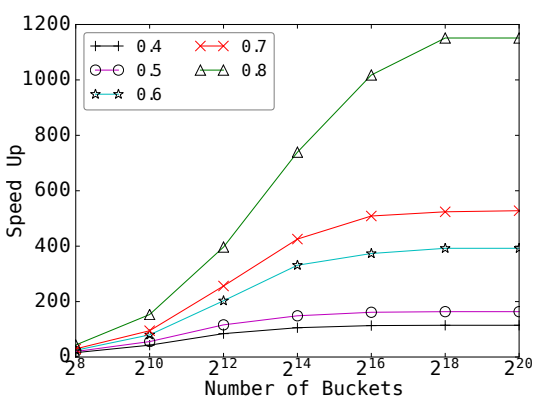


Figure 6.37: Speed-up – Cab

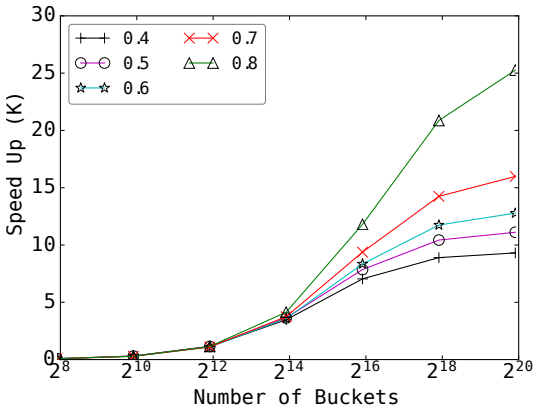


Figure 6.38: Speed-up – SM

Figure 6.39: Speed-up as a function of the bucket size

12 hours) the speed up reaches 202×. The maximum speed-up achieved for this dataset is 332×, preserving 86% of the $F1$-Score.

Figures 6.34 and 6.35 present the same experiment for the SM dataset. While we observe similar behaviour for the data points with low spatial detail, we also observe that the increase in the speed-up starts earlier and is steeper when the spatial detail is increased. This is because the SM dataset has lower geographic and temporal skew. If we observe the same data point as we did with the Cab dataset, we observe more pronounced speed-ups: For the spatial detail value of 16 and temporal step size of 48, LSH brings 1177× speed-up preserving 91% of the $F1$-Score. Next, we show that the maximum reachable speed-up is much higher when the number of buckets is increased.

6.2.3.2 Effect of the LSH Parameters

Figure 6.39 plots the speed-up as a function of the number of hash buckets. Different series represent different LSH similarity thresholds. We set the spatial detail and temporal step size of the signature calculation to 16 and 48, respectively.

Intuitively, $F1$-Score is not effected by the number of buckets. This is because if two entities have at least one identical band in their signatures, they are hashed

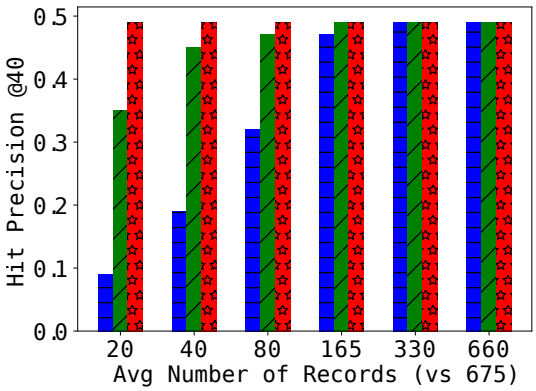


Figure 6.40: Hit precision @40

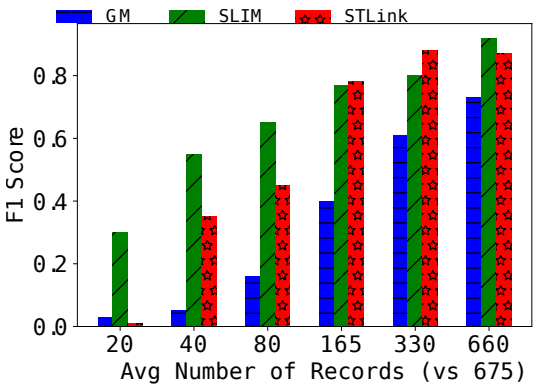


Figure 6.41: $F1$-Score

Figure 6.42: Comparison with existing work (Sub-figures a and b are sharing their legends

to the same bucket independent from the number of buckets. Yet, increased number of buckets increases the speed up as probability of hash collision decreases. Similarly, the LSH similarity threshold affects the relative $F1$-Score, as for smaller values of it the relative number of bands is increasing. Consequently, probability of becoming a candidate pair increases. Similarly, setting a large threshold requires signatures to be more similar in order to be hashed to the same bucket. At one extreme, threshold equals to 1 implies only a single band will be used and two signatures will be hashed to the same bucket only if they are identical.

We observe the increase in speed-up for the Cab dataset and the SM dataset in Figures 6.37 and 6.38, respectively. When the number of buckets is set to 2^{18} and the similarity threshold is set to 0.6, the speed-up becomes 380× with a relative $F1$-Score of 0.98 for the Cab dataset and $11,742\times$ with a relative $F1$-Score of 0.91 for the SM dataset. Since the number of entities in the SM dataset is much larger compared to the Cab dataset, we observe a significant difference in speed-up.

6.3 Comparison with Existing Work

In this section we first compare *ST-Link* with Stanford Entity Resolution Framework (SERF). SERF implements the R-Swoosh [24] algorithm. Next, we compare

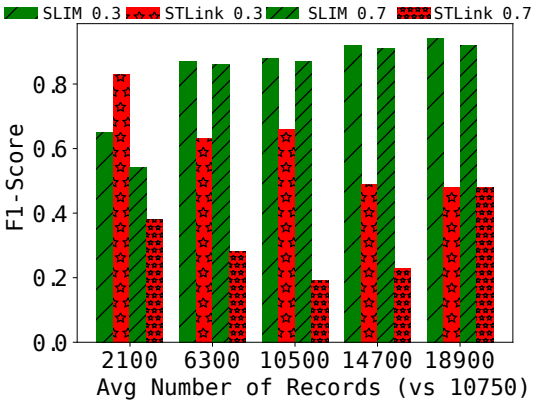


Figure 6.43: $F1$-Score

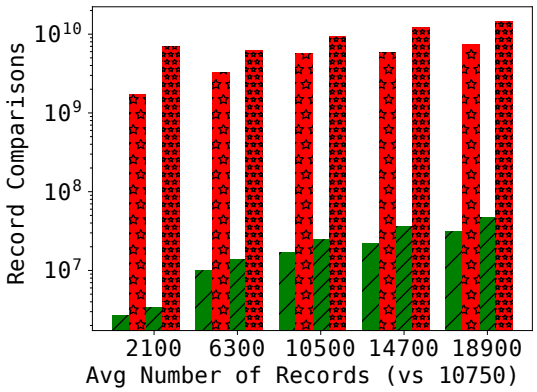


Figure 6.44: Record comparisons

Figure 6.45: Comparison with existing work (Sub-figures c and d are sharing their legends)

SLIM and *ST-Link* algorithms with each other and with another approach named GM. GM works by learning mobility models from entity records using Gaussian Mixtures and Markov-based models [19].

6.3.1 St-Link vs Serf

In addition to evaluating *ST-Link* under different settings, we also attempted to integrate our linkage model with the Stanford Entity Resolution Framework (SERF). SERF implements the R-Swoosh [24] algorithm. For this integration, users are arranged as entities and their events are considered as attributes. Given two entities, if they have enough number of co-occurring attributes satisfying the k-l diversity model, they are marked as a match.

Starting with pairwise comparison of entities, R-Swoosh algorithm gradually decreases the number of entities by merging the matching records, and deleting the dominated ones. While this is an effective method to decrease the number of comparisons on match heavy datasets, for datasets that contain few matching entities, the run-time is still $\mathcal{O}(N^2)$. Merging of two records is valid only when there is merge associativity between records. Given three records, r_1, r_2, and r_3,

comparisons of $\langle r_1, \langle r_2, r_3 \rangle \rangle$ and $\langle \langle r_1, r_2 \rangle, r_3 \rangle$ may result in different linkage decisions [46]. To alleviate this problem, the SERF framework also implements the Koosh algorithm [46]. Different than the R-Swoosh algorithm, when the Koosh algorithm finds a matching pair of entities, it does not merge them immediately, unless confidence is above a threshold. However, defining the confidence to use our spatio-temporal linkage model in SERF is not straightforward and requires further research, which we leave as future work.

Applying Koosh algorithm without using merges is almost brute force and using a small subset our dataset (15,268 users, 1,956,734 events, in total), SERF takes more than 50 hours of processing time in the same setting. In comparison, our algorithm finds the matching users in the same dataset in 30 minutes.

6.3.2 SLIM vs ST-Link vs GM

We compare *SLIM* and *ST-Link* with GM[4], which works by learning mobility models from entity records using Gaussian Mixtures and Markov-based models [19]. These models are later used for estimating the missing locations of users, and also setting weights to spatio-temporally close record pairs. While we only check records those are in the same temporal window, they also award pair of records those are from different temporal windows.

In this work, we do not compare our approach with other existing approaches because in its paper GM outperforms eight other existing approaches, excluding *ST-Link* and *SLIM*.

Hit Precision @k is used to measure the quality of ranking algorithms. It is calculated independently for all entities via the formula $1 - max((rank/k), 1)$ and averaged. The *rank* is the order of the true link in the list of all entities from the opposite dataset, sorted in decreasing order of their similarity score. Figures 6.40 and 6.41 show the Hit Precision @40 and $F1$-Score for three algorithms as a function of the average number of records. Since GM does not implement

[4]We used the code from the authors: `https://tinyurl.com/yagfaz5n`

any mechanisms to scale to a large number of records, to include it in our results, we took a 1 week subset of the data. The pivot dataset has 265 taxis with 675 records on average. We sampled 5 other datasets, with changing number of average number of records, ranging from 20 to 660. These datasets have 265 taxis, 133 of which are common with the pivot. With this setting the best achievable hit precision is 0.5.

From these two figures, we observe that hit precision values for GM algorithm is increasing as the average number of records increases. ST-Link reaches its maximum hit precision with as small as 20 records. *SLIM* outperforms GM in all data points, and reaches its best hit precision when the average number of records is 165. While all three algorithms are able to provide perfect hit precision @40, their performance in terms of $F1$-Score differs dramatically. When the average number of records is 20, *SLIM* reaches an $F1$-Score of 0.3, while the other two alternatives stay around 0.05. Since GM does not link entities with a single entity from the opposite dataset, we apply our linkage and stop threshold algorithm over their similarity scores. While ST-Link is able to rank true positive pairs at the top (we could conclude this from perfect hit precision), it is not always able to detect correct k and l values and resolve ambiguity. When the number of average records per entity increases to 660, we observe that *SLIM* still performs the best in terms of the $F1$-Score with 0.92. For the same data point, ST-Link and GM has $F1$-Scores of 0.87 and 0.73, respectively.

Since GM does not scale to larger datasets and ST-Link is the best competitor in terms of accuracy, we rule out GM in the further experiments. Figures 6.43 and 6.44 show $F1$-Score and number of pairwise record comparisons for different record densities, respectively. Green bars correspond to the *SLIM* algorithm, the red bars to the ST-Link algorithm. To improve the detail of the comparison, we use two different intersection amount ratios for each data point, 0.3 and 0.7. Bars with single hatches show the results for intersection amount ratio 0.3 and bars with double hatches show that of 0.7. We observe that *SLIM* outperforms ST-Link in terms of $F1$-Score in all data points except one and accuracy of ST-Link algorithm decreases when average number of records per entity increases. This is because *SLIM* is more robust to alibi record pairs than ST-Link,

even when alibi threshold mechanism of ST-Link is used. In this experiment, we set alibi threshold count to 3 for ST-Link. While larger alibi threshold values might give better accuracy for ST-Link as it is shown in their paper, increasing this threshold also increases the number of pairwise record comparisons.

Figure 6.44 shows that *SLIM* already makes three orders of magnitude less pairwise record comparisons than ST-Link. This is because our LSH-based scalability technique is more effective and ST-Link does not scale well in spatially dense areas.

6.4 Summary

In the experimental study, we first evaluated various aspects of the k-l diversity based linkage model and the *ST-Link* algorithm. We studied the scalability of the algorithm and showed that it scales linearly with the dataset size. We studied the effectiveness of the linkage and showed that high precision can be achieved. Using the unweighted version of our model, some of that precision can be traded off in order to achieve better recall values as well.

We also added the alibi threshold experiment to better observe the behaviour of the system. This threshold value is the number of alibi events can be tolerated before disregarding an entity pair. Our experiments showed that, setting a large threshold, t, increases the run-time of the algorithm, since there are possibly more candidate pairs. On the other hand, larger thresholds might also increase the recall. This is because, when there is inaccurate information in the data, disregarding an entity pair with a single alibi might harm the accuracy of the linkage.

We then evaluated similarity-based linkage and *SLIM* algorithm in terms of the quality of linkage, robustness under changing spatio-temporal levels, record inclusion probability and entity intersection ratios of two mobility datasets. We

observe that *SLIM* is robust to spatio-temporal level of the input data and to different values of workload parameters. One interesting finding is while increasing the level of detail after a certain point does not contribute to the precision and recall, it harms the scalability of the system as it increases the number of record comparisons. This is expected, as the aggregation performed on a per-window basis is critical in terms of scalability.

Moreover, we also measured the scalability of this approach by comparing the number of pairwise entity comparisons and the $F1$-Score of the resulting linkage, by varying the parameters of the LSH procedure for mobility histories, i.e., the similarity threshold t, the query time window size, and the spatial level of the dominating grid cells. Depending on the values select for LSH parameters we observe it could provide $380\times$ speed-up with a relative $F1$-Score of 0.98 for the Cab dataset and $11,742\times$ with a relative $F1$-Score of 0.91 for the SM dataset. Since the number of entities in the SM dataset is much larger compared to the Cab dataset, we observe a significant difference in speed-up.

Lastly, we compared our proposed approaches, *ST-Link* and *SLIM* with each other and with one existing work in the literature, GM [19]. We did not compare our approach with other existing approaches because in its paper GM outperforms eight other existing approaches. Our experiments show that GM does not scale to larger datasets and *ST-Link* and *SLIM* are the best competitors, comparable to each other, in terms of accuracy. We observe both approaches could perform highly accurate linkage while *SLIM* proving even more accurate linkage by performing three orders of magnitude less pair-wise comparisons.

Chapter 7

Literature Review

In this chapter we give a literature review. The most relevant problem to our work in the literature is user identity linkage. In addition to user identity linkage, there are six lines of related work; namely record linkage, temporal record linkage and entity evolution, spatial record linkage and spatial joins, trajectory join, user identification and trajectory de-anonymization.

Record Linkage. One of the earliest appearances of the term *record linkage* is by Newcombe et al. [25, 47]. In the literature, it is also referred to as entity resolution (ER), deduplication, object identification, and reference reconciliation. Several surveys exist on the topic [15, 16, 17]. Most of the work in this area focus on a single type of databases and define the linked records with respect to a similarity metric. The input to such a record linkage algorithm is a set of records and the output from it is a clustering of records. In contrast, our problem involves linking users from two datasets, where each user can have multiple spatio-temporal records. While many work on record linkage focus on accuracy [48, 49, 14] and a few on scalability [50], our work must consider both. In our case, successful linkage does not rely solely on the similarity of records and as such our proposed solutions search multiple diverse matches, aka k-l diversity, and also makes sure that there are no negative matches, aka alibis. To the best of our knowledge, this is a novel approach for record linkage, specifically targeted at spatio-temporal

datasets.

Temporal Record Linkage and Entity Evolution. Temporal record linkage differs from traditional record linkage in that it takes entity evolution into account (e.g., a person can change her phone number). The *time decay* model captures the probability of an entity changing its attribute value within a given time interval [51]. The *mutation model* learns the probability of an attribute value re-appearing over time [52]. The *transition model* learns the probability of complex value transitions over time [53]. Furthermore, declarative rules can be used to link records temporally [54]. Transition model can also capture complex declarative rules. Temporal record linkage algorithms are able to capture the entity evolution and determine if an entity has changed the value of one or more of its attributes. Our problem has some resemblance to entity evolution, since the location attributes of the users change over time. However, this change can be better described as entity mobility, rather than entity evolution. Application of aforementioned models to spatio-temporal datasets might be effective in predicting a user's next stop or calculating the probability of whether a user will return back to a given location. Yet, they would fell short of linking spatio-temporal records of users.

Spatial Record Linkage and Spatial Joins. Many join and self-join algorithms are proposed in the literature for spatial data [55]. Sehgal et al. [56] proposes a method to link the spatial records by integrating spatial and non-spatial (e.g. location name) features. However, spatial record linkage and spatial join algorithms are not extensible to spatio-temporal data as they are based on intersection of minimum bounding boxes, one-sided nearest join, or string similarity. Spatio-temporal joins are more complex with constrains on both spatial and temporal domains [57]. Yet our problem involves more than spatio-temporal records, it involves matching spatio-temporal record series from two datasets.

Trajectory Join. Bakalov et al. [58] define the trajectory joins as the identification of all pairs of similar trajectories given two datasets. They represent an object trajectory as a sequence of symbols. Based on the symbol similarity, they prune the pairwise trajectory comparisons. Effective evaluation of symbol similarity is supported by a tree-like index scheme. In [57], the authors extend

the problem to continuous queries over streaming spatio-temporal trajectories. An important difference between trajectory join algorithms and our work is that trajectory similarity is not necessarily an indication of a linkage and vice verse. If one of the datasets is denser than the other, trajectories would be dissimilar, yet we still can have matching user pairs based on k-l linkage. However, some indexing structures of trajectory join algorithms are closely related to our approach. There are multiple indexing schemes for spatio-temporal data. In [59, 60, 61, 62] various grid based structures are used for indexing. Our spatial filtering approach is similar in its use of a grid-based index, but instead of associating objects with grid cells, we associate users with grid cells based on the frequency of their events residing in these cells. There are also tree-like spatio-temporal indexing structures, surveyed in [63]. A common theme of these works is the reduction of the update cost, which is not a concern in our work.

User Identification. Our work has commonalities with the work done in the area of user identification. For instance, de Montjoye et al. [1] has shown that, given a spatio- temporal dataset of call detail records, one can uniquely identify the 95 % of the population by using 4 randomly selected spatio-temporal points. Similar to our discussion, the authors mention that spatio-temporal points do not contribute to information gain equally. In our work, we cover this by introducing a *weight* function. Unlike our work, [1] does not consider the linkage problem, that is matching users from two spatio-temporal record sets. instead, they study how users can be uniquely identified within a single dataset using a small subset of their records. Another related work is [64], in which authors show that using the credit card metadata, they can identify unique users and group the transactions with respect to users. In addition to spatio-temporal reference data, they use the transaction price and gender as auxiliary information. Another related work is from Rossi et al. [65], in which user identification techniques for GPS mobility data is presented. They use a classification based algorithm rather than pairwise comparison of records. Importantly, our algorithm does not use any auxiliary information but only spatio-temporal data, and it aims to match entities across datasets. Our goal is not the identification of the users in a single dataset but matching them across datasets.

User Identity Linkage. Many of the attempts in user identity linkage (aka

user reconciliation, account linkage as surveyed in [66]) utilize additional information such as the network graph [67, 68, 69], user profile information (such as usernames or photos) [70, 71, 72], semantic information related to locations [73], or a combination of these [74, 75]. They mainly differ by the information used to perform the linkage or by the definition of similarity measures among entities. A small number of these work also discuss the scalability challenges [18, 76]. However, in many cases, only the spatio-temporal information is present in mobility data, and many of the other identifiers are likely to be anonymized. Use of only spatio-temporal information also fits better with purpose restriction and minimal data collection with consent. Therefore, we focus on linkage using only the spatio-temporal information.

An important challenge when only spatio-temporal information is utilized is to define similarity measures of entities and records (surveyed in[77, 78]). Some work express this similarity based on densities of location histories of entities [79, 80]. They work either by matching user histograms [80], or using the frequencies of visits to specific locations during specific times [81]. Statistical learning approaches are also used to relate social media datasets with Call Detail Records [82, 83]. However, these algorithms depend on discriminative patterns of entities, which is not likely to be present in many datasets, such as the Cab dataset we use in this work.

There have been studies to define the similarity among entities using the co-occurrences of their records [18, 76, 14, 19]. These are the most similar group of work to ours. *ST-Link* and *SLIM* are shown to outperform existing work in terms of accuracy and scalability. Among these algorithms, some model the user mobility to handle data sparsity issues and also to compute a similarity score [19, 14]. For example, in [14], it is assumed that the number of visits of each user to a location during a time period follows Poisson distribution and records on each service are independent from each other following Bernoulli distribution. There are two limitations compared to our work. First, these work do not address scalability. Among the contributions we make is to utilize locality-sensitive hashing to perform linkage in a scalable fashion and also use spatial and temporal filtering steps to decrease the number of candidate pairs. Second, the related work assume

that the entities from one of the datasets is a subset of the other. In our work, we address the problem of linkage stop condition, thus avoiding this assumption.

Cao et al. [18] use co-occurrences of the records to measure similarity among entities. The strength of the co-occurrences is defined inversely proportional with the frequency of locations. A multi-resolution filtering step is developed for scalability. Different from our approach, the data is pre-processed to add semantic information to locations. They use time to transform each trajectory to a set of stay points, i.e., the location points that the user stay for a while. Unlike ours, they do not define a concept of dissimilarity, called *alibi* in our work. Furthermore, they do not automatically determine a similarity threshold to stop linkage, called rejection in their case, and set it manually.

Finally, the related work we have discussed on mobility history linkage, including our own work, is different than trajectory similarity-based approaches. Trajectory similarity is measured using subsequence similarity metrics such as the length of the longest common subsequence, Frechet distance, dynamic time warping, and edit distance [84]. There are also trajectory distinguishing techniques that include trajectory specific information like speed, acceleration, and direction of movement [85]. As opposed to our work, these approaches fell short in addressing asynchrony of the datasets and capturing alibi event pairs. Our approach is more generic as it depends on less features when computing the similarity scores.

Chapter 8

Conclusion

In this work, we studied matching the spatio-temporal usage records belonging to the same real-world entity across records from different location-enhanced services. This is a fundamental step towards unifying mobility datasets.

The first approach we developed is rule-based linkage, based on the concept of k-l diversity. By introducing the k-l–diversity model, a novel concept that captures both spatial and temporal diversity aspects of the linkage, we study the challenge of defining proximity between usage records of entities from different datasets. As part of this model, we introduced the concept of an *alibi*, which effectively filters out negative matches and significantly improves the linkage quality.

To realize the k-l–diversity model, we developed the scalable *ST-Link* algorithm that makes use of effective filtering steps. Taking advantage of the spatial nature of the data, users are associated with *dominating grids* — grids that contain most activities of their entities. This enables processing each grid independently, improving scalability. Taking advantage of the temporal nature of the data, we slide a window over both datasets jointly and maintain set of candidate users that have co-occurring events but no alibis. The set of candidate entities are pruned as the window is slided.

Next we developed another model, which we call similarity-based linkage. For

this, we first developed a summary representation of mobility records of the entities and a novel way to compute a similarity score among these summaries. This score captures the closeness in time and location of the records, while not penalizing temporal asynchrony. Moreover, it captures the concept of alibi as well, which greatly improves both efficiency and accuracy.

We applied a bipartite matching process to identify the final linked entity pairs, using a stop similarity threshold for the linkage. This threshold is determined by fitting a mixture model over similarity scores to minimize the expected $F1$-Score metric. We also addressed the scalability challenge and employed a locality-sensitive hashing (LSH) approach for mobility histories, which avoids unnecessary pairwise comparisons. To realize effectiveness of the techniques in practice, we implemented an algorithm called *SLIM*.

Our experimental evaluation, conducted with several data sets showed that the running time of the *ST-Link* algorithm scales linearly with the dataset size. Moreover, precision of the linkage results is practically 1 for most k-l settings. Likewise, *SLIM* outperforms two existing approaches, including *ST-Link* in terms of accuracy and scalability. Moreover, LSH brings two to four orders of magnitude speed-up to the linkage in our experimental settings.

As most data science tasks require large amount of data for accurate training with higher confidence, scientists need to combine data from multiple sources to produce accurate aggregate patterns. The goal of this work is to gather mobility data from multiple sources and merge them into a single one. This unification enables data scientists to obtain information that they cannot derive by mining only one set of usage records. Yet, there are still many questions to be answered in the way towards unifying mobility datasets. For example, merging the usage histories of the linked entities in a way that it satisfies ICAR properties and development of a generic and distributed linkage framework are left as the future work.

Bibliography

[1] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, 2013.

[2] A. Skovsgaard, D. Sidlauskas, and C. Jensen, "Scalable top-k spatio-temporal term querying," in *IEEE Int. Conference on Data Engineering (ICDE)*, pp. 148–159, March 2014.

[3] A. Pentland, D. Lazer, D. Brewer, and T. Heibeck, "Using reality mining to improve public health and medicine," *Studies in health technology and informatics*, vol. 149, pp. 93–102, 02 2009.

[4] A. Pentland, *Reality Mining of Mobile Communications: Toward A New Deal On Data*, pp. 1–1. Boston, MA: Springer US, 2009.

[5] Y. Zheng, "Methodologies for cross-domain data fusion: An overview," *IEEE Transactions on Big Data*, vol. 1, pp. 16–34, March 2015.

[6] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, pp. 32–39, November 2011.

[7] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani, "The role of the airline transportation network in the prediction and predictability of global epidemics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 7, pp. 2015–2020, 2006.

[8] D. O. Rodrigues, A. Boukerche, T. H. Silva, A. A. Loureiro, and L. A. Villas, "Combining taxi and social media data to explore urban mobility issues," *Computer Communications*, vol. 132, pp. 111 – 125, 2018.

[9] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, pp. 29:1–29:41, May 2015.

[10] J. E. Steele, P. R. Sundsøy, C. Pezzulo, V. A. Alegana, T. J. Bird, J. Blumenstock, J. Bjelland, K. Engø-Monsen, Y.-A. de Montjoye, A. M. Iqbal, K. N. Hadiuzzaman, X. Lu, E. Wetter, A. J. Tatem, and L. Bengtsson, "Mapping poverty using mobile phone and satellite data," *Journal of The Royal Society Interface*, vol. 14, no. 127, 2017.

[11] D. Hristova, M. J. Williams, M. Musolesi, P. Panzarasa, and C. Mascolo, "Measuring urban social diversity using interconnected geo-social networks," in *Proceedings of the 25th Int. Conference on World Wide Web*, International Conference on World Wide Web (WWW), (Republic and Canton of Geneva, Switzerland), pp. 21–30, Int. World Wide Web Conferences Steering Committee, 2016.

[12] H. Wang, Y. Li, G. Wang, and D. Jin, "You are how you move: Linking multiple user identities from massive mobility traces," in *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 189–197, SIAM, 2018.

[13] F. Basık, "Scalable linkage across location enhanced services.," in *Proceedings of the Very Large Databases Conference (VDLB)*, PhD Workshop, 2017.

[14] C. Riederer, Y. Kim, A. Chaintreau, N. Korula, and S. Lattanzi, "Linking users across domains with location data: Theory and validation," in *Proc. of the 25th Int. Conf.on World Wide Web*, pp. 707–719, 2016.

[15] P. Christen, *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012.

[16] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, no. 1, pp. 1–16, 2007.

[17] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proceedings of the VLDB Endowment*, vol. 3, pp. 484–493, Sept. 2010.

[18] W. Cao, Z. Wu, D. Wang, J. Li, and H. Wu, "Automatic user identification method across heterogeneous mobility data sources," in *IEEE Int. Conference on Data Engineering (ICDE)*, pp. 978–989, 2016.

[19] H. Wang, C. Gao, Y. Li, G. Wang, D. Jin, and J. Sun, "De-anonymization of mobility trajectories: Dissecting the gaps between theory and practice," in *The 25th Annual Network & Distributed System Security Symposium (NDSS'18)*, 2018.

[20] F. Xu, Z. Tu, Y. Li, P. Zhang, X. Fu, and D. Jin, "Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data," in *International Conference on World Wide Web (WWW)*, WWW '17, (Republic and Canton of Geneva, Switzerland), pp. 1241–1250, 2017.

[21] F. Basık, B. Gedik, C. Etemoğlu, and H. Ferhatosmanoğlu, "Spatio-temporal linkage over location-enhanced services," *IEEE Transactions on Mobile Computing*, vol. 17, pp. 447–460, Feb 2018.

[22] L. Getoor and A. Machanavajjhala, "Entity resolution: Theory, practice & open challenges," in *VLDB Conference*, 2012.

[23] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, (New York, NY, USA), pp. 604–613, ACM, 1998.

[24] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, "Swoosh: A generic approach to entity resolution," *The VLDB Journal*, vol. 18, pp. 255–276, Jan. 2009.

[25] H. B. Newcombe and J. M. Kennedy, "Record linkage: Making maximum use of the discriminating power of identifying information," *Commun. ACM*, vol. 5, pp. 563–566, Nov. 1962.

[26] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," in *ACM (SIGMOD)*, pp. 127–138, ACM, 1995.

[27] S. E. Whang, D. Marmaros, and H. Garcia-Molina, "Pay-as-you-go entity resolution," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 25, no. 5, 2013.

[28] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB Conference*, pp. 518–529, 1999.

[29] P. Christen and R. Gayler, "Towards scalable real-time entity resolution using a similarity-aware inverted index approach," in *Proceedings of the 7th Australasian Data Mining Conference - Volume 87*, pp. 51–60, Australian Computer Society, Inc., 2008.

[30] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[31] H. Samet, J. Sankaranarayanan, and M. Auerbach, "Indexing methods for moving object databases: Games and other applications," in *ACM (SIGMOD)*, pp. 169–180, ACM, 2013.

[32] S. Ghemawat and J. Dean, "LevelDB." https://github.com/google/leveldb, 2015.

[33] J. S. Whissell and C. L. Clarke, "Effective measures for inter-document similarity," in *ACM International Conference on Information Knowledge Management (CIKM)*, (New York, NY, USA), pp. 1361–1370, ACM, 2013.

[34] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, "Okapi at trec-3," in *TREC '94: The Third Text REtrieval Conference*, pp. 109–126, 1996.

[35] H. W. Kuhn and B. Yaw, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[36] J. M. Kurtzberg, "On approximation methods for the assignment problem," *J. ACM*, vol. 9, pp. 419–439, Oct. 1962.

[37] D. A. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, 2009.

[38] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[39] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171, June 2011.

[40] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *CoRR*, vol. abs/1408.2927, 2014.

[41] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press, 2011.

[42] R. Corless, G. Gonnet, D. E. G. Hare, D. Jeffrey, and D. E. Knuth, "On the lambert w function," *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 01 1996.

[43] M. Sadinle and S. E. Fienberg, "A generalized fellegi–sunter framework for multiple record linkage with application to homicide record systems," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 385–397, 2013.

[44] A. Saeedi, E. Peukert, and E. Rahm, "Comparative evaluation of distributed clustering schemes for multi-source entity resolution," in *Advances in Databases and Information Systems* (M. Kirikova, K. Nørvåg, and G. A. Papadopoulos, eds.), (Cham), pp. 278–293, Springer International Publishing, 2017.

[45] L. Fegaras, "A new heuristic for optimizing large queries," in *Database and Expert Systems Applications* (G. Quirchmayr, E. Schweighofer, and T. J.

Bench-Capon, eds.), (Berlin, Heidelberg), pp. 726–735, Springer Berlin Heidelberg, 1998.

[46] O. Benjelloun, H. Garcia-Molina, H. Kawai, T. E. Larson, D. Menestrina, Q. Su, S. Thavisomboon, and J. Widom, "Generic entity resolution in the serf project," Technical Report 2006-14, June 2006.

[47] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James, "Automatic linkage of vital records: Computers can be used to extract "follow-up" statistics of families from files of routine records," *Science*, vol. 130, no. 3381, pp. 954–959, 1959.

[48] X. Dong, A. Halevy, and J. Madhavan, "Reference reconciliation in complex information spaces," in *ACM (SIGMOD)*, pp. 85–96, ACM, 2005.

[49] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 5, 2007.

[50] S. E. Whang and H. Garcia-Molina, "Joint entity resolution on multiple datasets," *The VLDB Journal (VLDBJ)*, vol. 22, no. 6, pp. 773–795, 2013.

[51] P. Li, X. Dong, A. Maurino, and D. Srivastava, "Linking temporal records," *VLDB Conference*, vol. 4, no. 11, pp. 956–967, 2011.

[52] Y.-H. Chiang, A. Doan, and J. F. Naughton, "Modeling entity evolution for temporal record matching," in *ACM (SIGMOD)*, pp. 1175–1186, 2014.

[53] F. Li, M. L. Lee, W. Hsu, and W.-C. Tan, "Linking temporal records for profiling entities," in *ACM (SIGMOD)*, pp. 593–605, ACM, 2015.

[54] D. Burdick, M. A. Hernández, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. Stanoi, S. Vaithyanathan, and S. R. Das, "Extracting, linking and integrating data from public sources: A financial case study.," *IEEE Data Eng. Bull.*, vol. 34, no. 3, pp. 60–67, 2011.

[55] E. H. Jacox and H. Samet, "Spatial join techniques," *ACM Trans. Database Syst.*, vol. 32, Mar. 2007.

[56] V. Sehgal, L. Getoor, and P. D. Viechnicki, "Entity resolution in geospatial data integration," in *Proceedings of the 14th Annual ACM Int. Symposium on Advances in Geographic Information Systems*, pp. 83–90, 2006.

[57] P. Bakalov and V. Tsotras, "Continuous spatiotemporal trajectory joins," in *GeoSensor Networks*, vol. 4540 of *Lecture Notes in Computer Science*, pp. 109–128, Springer Berlin Heidelberg, 2008.

[58] P. Bakalov, M. Hadjieleftheriou, E. Keogh, and V. J. Tsotras, "Efficient trajectory joins using symbolic representations," in *Int. Conference on Mobile Data Management (MDM)*, pp. 86–93, 2005.

[59] B. Gedik and L. Liu, "Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system," in *Advances in Database Technology - EDBT 2004*, vol. 2992 of *Lecture Notes in Computer Science*, pp. 67–87, Springer Berlin Heidelberg, 2004.

[60] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou, "Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring," in *ACM (SIGMOD)*, pp. 634–645, ACM, 2005.

[61] J. M. Patel, Y. Chen, and V. P. Chakka, "Stripes: An efficient index for predicted trajectories," in *ACM (SIGMOD)*, pp. 635–646, ACM, 2004.

[62] M. F. Mokbel, X. Xiong, and W. G. Aref, "Sina: Scalable incremental processing of continuous queries in spatio-temporal databases," in *ACM (SIGMOD)*, pp. 623–634, ACM, 2004.

[63] Y. Manolopoulos, Y. Theodoridis, and V. Tsotras, "Spatiotemporal access methods," in *Advanced Database Indexing*, vol. 17 of *Advances in Database Systems*, pp. 141–166, Springer US, 2000.

[64] Y.-A. de Montjoye, L. Radaelli, V. K. Singh, *et al.*, "Unique in the shopping mall: On the reidentifiability of credit card metadata," *Science*, vol. 347, no. 6221, pp. 536–539, 2015.

[65] L. Rossi, J. Walker, and M. Musolesi, "Spatio-temporal techniques for user identification by means of GPS mobility data," *CoRR*, vol. abs/1501.06814, 2015.

[66] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *SIGKDD Explorations*, vol. 18, pp. 5–17, 2016.

[67] N. Korula and S. Lattanzi, "An efficient reconciliation algorithm for social networks," *VLDB Conference*, vol. 7, pp. 377–388, 2014.

[68] J. Liu, F. Zhang, X. Song, Y.-I. Song, C.-Y. Lin, and H.-W. Hon, "What's in a name: an unsupervised approach to link users across communities," in *WSDM*, 2013.

[69] N. Vesdapunt and H. Garcia-Molina, "Identifying users in social networks with limited information," in *IEEE Int. Conference on Data Engineering (ICDE)*, pp. 627–638, 2015.

[70] O. Goga, P. Loiseau, R. Sommer, R. Teixeira, and K. P. Gummadi, "On the reliability of profile matching across large online social networks," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1799–1808, 2015.

[71] X. Mu, F. Zhu, E.-P. Lim, J. Xiao, J. Wang, and Z.-H. Zhou, "User identity linkage by latent user space modelling," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[72] V. Sharma and C. Dyreson, "Linksocial: Linking user profiles across multiple social media platforms," in *2018 IEEE International Conference on Big Knowledge (ICBK)*, pp. 260–267, Nov 2018.

[73] S. Liu, S. Wang, and F. Zhu, "Structured learning from heterogeneous behavior for social identity linkage," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 27, no. 7, pp. 2005–2019, 2015.

[74] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan, "Hydra: large-scale social identity linkage via heterogeneous behavior modeling," in *ACM (SIGMOD)*, 2014.

[75] Y. Wang, C. Feng, L. Chen, H. Yin, C. Guo, and Y. Chu, "User identity linkage across social networks via linked heterogeneous network embedding," *World Wide Web*, pp. 1–22, Apr 2018.

[76] D. Kondor, B. Hashemian, Y.-A. de Montjoye, and C. Ratti, "Towards matching user mobility traces in large-scale datasets," *CoRR*, vol. abs/1709.05772, 2017.

[77] Y. Kanza, E. Kravi, E. Safra, and Y. Sagiv, "Location-based distance measures for geosocial similarity," *ACM Transactions on Web*, vol. 11, pp. 17:1–17:32, July 2017.

[78] H. Wang, H. Su, K. Zheng, S. W. Sadiq, and X. Zhou, "An effectiveness study on trajectory similarity measures," in *Australian Database Confrence*, pp. 13–22, 2013.

[79] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *International Conference on World Wide Web (WWW)*, 2013.

[80] J. Unnikrishnan and F. M. Naini, "De-anonymizing private data by matching statistics," *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1616–1623, 2013.

[81] L. Rossi and M. Musolesi, "It's the way you check-in: identifying users in location-based social networks," in *International Conference on Online Social Networks*, 2014.

[82] A. Cecaj, M. Mamei, and N. Bicocchi, "Re-identification of anonymized cdr datasets using social network data," in *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pp. 237–242, 2014.

[83] A. Cecaj, M. Mamei, and F. Zambonelli, "Re-identification and information fusion between anonymized cdr and social network data," *Journal of Ambient Intelligence and Humanized Computing*, vol. 7, pp. 83–96, Feb 2016.

[84] G. Atluri, A. Karpatne, and V. Kumar, "Spatio-temporal data mining: A survey of problems and methods," *ACM Comput. Surv.*, pp. 83:1–83:41, 2018.

[85] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Distinguishing trajectories from different drivers using incompletely labeled trajectories," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pp. 863–872, 2018.